### 

**BVRIT HYDERABAD**

**College of Engineering for Women**

Department of Computer Science & Engineering

**Computer Networks**

**Lab Manual**

**III B.TECH - I SEM**

**regulation : r 22**



**BVRIT HYDERABAD**

**College of Engineering for Women**

**Rajiv Gandhi Nagar, Bachupally, Hyderabad -90**

**Department of Computer Science & Engineering**

**BVRIT HYDERABAD**

**College of Engineering for Women**

**Rajiv Gandhi Nagar, Bachupally, Hyderabad -90**

**Computer Networks Lab**

**III Year – I Semester**

**INSTITUTE VISION & MISSION**

**VISION**

To emerge as the best among the institutes of technology and research in the country dedicated to the cause of promoting quality technical education.

**MISSION**

At BVRITH, we strive to

* Achieve academic excellence through innovative learning practices.
* Enhance intellectual ability and technical competency for a successful career.
* Encourage research and innovation.
* Nurture students towards holistic development with emphasis on leadership skills. life skills and human values.

**DEPARTMENT VISION & MISSION**

**VISION**

Develop women as technocrats, researchers and entrepreneurs in the field of computer science and engineering.

**MISSION**

M1: To impart quality education in Computer Science and Engineering by means of learning techniques and value-added courses.

M2: To inculcate professional excellence and research culture by encouraging projects in cutting-edge technologies through industry interactions.

M3: To build leadership skills, ethical values and teamwork among the students.

M4: To strengthen the collaboration of department and industry through internships, mentorships and professional body activities.

**Program Educational Objectives (PEOs)**

PEO-1: Adapt emerging technologies to contribute to the technical innovations for the progressive development in their respective fields.

PEO-2: Productively engage in multidisciplinary research areas by applying the basic principles of engineering sciences.

PEO-3: Demonstrate strong technical skills to bring out novel designs/products to address social & environmental issues.

PEO-4: Exhibit professional attitude, teamwork and practice code of ethics.

**Program Specific Objectives (PSOs)**

PSO 1 : Ability to apply learned skills to build optimized solutions pertaining to Computer & Communication Systems, Data Processing and Artificial Intelligence.

PSO 2 Employ standard strategies and practices in project development using FOSS (Free Open Source Software)

**Program Outcomes (POs)**

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science,engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis: i**dentify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques,resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as,being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

**Course Outcomes**

|  |  |  |
| --- | --- | --- |
| **COMPUTER NETWORKS LAB** | | **Blooms Level** |
| After this course, students shall be able to: | |  |
| C504.1 | Implement various data link layer farming methods and error detection mechanisms | Analyze |
| C504.2 | Design the shortest route between source and destination in the network. | Create |
| C504.3 | Design a broadcast tree for the given subnet and cipher text using DES algorithm and also decipher it. | Create |
| C504.4 | Create public key encryption to encode the given text using cryptography | Create |

Mapping of COs with Program Outcomes (POs) & Program Specific Outcomes (PSOs)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **COMPUTER NETWORKS LAB** | | | | | | | | | | | | | | | |
| **Course**  **Outcomes** | **Program Outcomes (PO)** | | | | | | | | | | | | | **PSOs** | |
|  | **PO1** | **PO2** | **PO3** | | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** |
| C504.1 | 3 | 2 | | - | - | - | - | - | 1 | - | - | - | - | 3 | 1 |
| C504.2 | 3 | 1 | | 3 | 3 | 1 | - | 2 | 1 | 2 | 2 | - | 2 | 3 | 2 |
| C504.3 | 2 | 1 | | 3 | 3 | - | 2 | 2 | 1 | 2 | 2 | - | 2 | 3 | 2 |
| C504.4 | 2 | 1 | | 3 | 3 | 1 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 3 | 2 |
| **Mean** | **2.5** | **1.25** | | **3** | **3** | **1** | **2** | **2** | **1** | **2** | **2** | **1** | **2** | **3** | **1.75** |

**Syllabus as per JNTU Hyderabad**

**List of Experiments**

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**

**III Year B .Tech CSE I- Sem L T P C**

**0 0 2 1**

**CS504PC: COMPUTER NETWORKS Lab**

**List of Experiments**

**Computer Networks Experiments**

1. Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.

2. Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP

3. Develop a simple data link layer that performs the flow control using the sliding window protocol,

and loss recovery using the Go-Back-N mechanism.

4. Implement Dijsktra’s algorithm to compute the shortest path through a network

5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.

6. Implement distance vector routing algorithm for obtaining routing tables at each node.

7. Implement data encryption and data decryption

8. Write a program for congestion control using Leaky bucket algorithm.

9. Write a program for frame sorting technique used in buffers.

10. Wireshark

i. Packet Capture Using Wire shark

ii. Starting Wire shark

iii. Viewing Captured Traffic

iv. Analysis and Statistics & Filters.

How to run Nmap scan

Operating System Detection using Nmap

Do the following using NS2 Simulator

i. NS2 Simulator-Introduction

ii. Simulate to Find the Number of Packets Dropped

iii. Simulate to Find the Number of Packets Dropped by TCP/UDP

iv. Simulate to Find the Number of Packets Dropped due to Congestion

v. Simulate to Compare Data Rate& Throughput.

vi. Simulate to Plot Congestion for Different Source/Destination

vii. Simulate to Determine the Performance with respect to Transmission of Packets

**BVRIT HYDERABAD**

**College of Engineering for Women**

**Rajiv Gandhi Nagar, Bachupally, Hyderabad -90**

### DO’S

* On entering the lab fill the details in log book.
* Shutdown the system properly while leaving the lab.
* Keep the Computer lab premises clean and tidy.
* Contact the System Administrator if you notice any kind of machine malfunction.

### DONT’S

* Don’t install any application software.
* Don’t disconnect or modify any machine, either PC or peripheral equipment.
* Don’t connect your personal laptop machine in places other than designated.
* Don’t eat or drink in the computer lab.
* Don’t carry your mobile phones.
* Don’t connect USB storage Devices to PC’s.

**BVRIT HYDERABAD**

**College of Engineering for Women**

**Rajiv Gandhi Nagar, Bachupally, Hyderabad -90**

**Department of Computer Science & Engineering**

**COMPUTER NETWORKS LAB**

**III Year – I Semester, CSE**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name of the Experiment** | **Page No.** |
| **Lab Cycle-1** | | |
| 1 | Implement the data link layer framing methods such as character, character-stuffing and bit stuffing. | **16** |
| 2 | Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP | **18** |
| 3 | Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism. | **25** |
| **Lab Cycle-2** | | |
| 4 | Implement Dijsktra’s algorithm to compute the shortest path through a network | **28** |
| 5 | Take an example subnet of hosts and obtain a broadcast tree for the subnet. | **33** |
| 6 | Implement distance vector routing algorithm for obtaining routing tables at each node. | **41** |
| **Lab Cycle-3** | | |
| 7 | Implement data encryption and data decryption | **45** |
| 8 | Write a program for congestion control using the Leaky bucket algorithm. | **48** |
| 9 | Write a program for frame sorting technique used in buffers. | **49** |
| **Lab Cycle-4** | | |
| 10 | Wireshark  i. Packet Capture Using Wire shark  ii. Starting Wire shark  iii. Viewing Captured Traffic  iv. Analysis and Statistics & Filters. | **54** |
|  | How to run Nmap scan | **64** |
|  | Operating System Detection using Nmap | **66** |
|  | Do the following using NS2 Simulator  i. NS2 Simulator-Introduction  ii. Simulate to Find the Number of Packets Dropped  iii. Simulate to Find the Number of Packets Dropped by TCP/UDP  iv. Simulate to Find the Number of Packets Dropped due to Congestion  v. Simulate to Compare Data Rate & Throughput.  vi. Simulate to Plot Congestion for Different Source/Destination  vii. Simulate to Determine the Performance with respect to Transmission of Packets | **67** |

Prepared by Verified by

**Course Coordinator-1**

**Course Coordinator - 2 Module Coordinator**

**BVRIT HYDERABAD**

**College of Engineering for Women**

**Rajiv Gandhi Nagar, Bachupally, Hyderabad -90**

**Department of Computer Science & Engineering**

**COMPUTER NETWORKS LAB**

**IIIYear – I Semester, CSE**

**STUDENT EVALUATION SHEET**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Name of the Experiment** | **Date of Expt** | **Viva (5M)** | **Program Execution**  **Marks (5M)** | **Record Marks (5M)** | **Total**  **Marks**  **(15 M)** | **Signature of Faculty Member** |
| 1 | Implement the data link layer framing methods such as character, character-stuffing and bit |  |  |  |  |  |  |
| 2 | Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP |  |  |  |  |  |  |
| 3 | Develop a simple data link layer that performs the flow control using the sliding window protocol,  and loss recovery using the Go-Back-N mechanism. |  |  |  |  |  |  |
| 4 | Implement Dijsktra’s algorithm to compute the shortest path through a network |  |  |  |  |  |  |
| 5 | Take an example subnet of hosts and obtain a broadcast tree for the subnet. |  |  |  |  |  |  |
| 6 | Implement distance vector routing algorithm for obtaining routing tables at each node. |  |  |  |  |  |  |
| 7 | Implement data encryption and data decryption |  |  |  |  |  |  |
| 8 | Write a program for congestion control using Leaky bucket algorithm. |  |  |  |  |  |  |
| 9 | Write a program for frame sorting technique used in buffers. |  |  |  |  |  |  |
| 10 | Wireshark  i. Packet Capture Using Wire shark  ii. Starting Wire shark  iii. Viewing Captured Traffic  iv. Analysis and Statistics & Filters. |  |  |  |  |  |  |
|  | Operating System Detection using Nmap |  |  |  |  |  |  |
|  | Do the following using NS2 Simulator  i. NS2 Simulator-Introduction  ii. Simulate to Find the Number of Packets Dropped  iii. Simulate to Find the Number of Packets Dropped by TCP/UDP  iv. Simulate to Find the Number of Packets Dropped due to Congestion  v. Simulate to Compare Data Rate& Throughput.  vi. Simulate to Plot Congestion for Different Source/Destination  vii. Simulate to Determine the Performance with respect to Transmission of Packets |  |  |  |  |  |  |
| **TOTAL** | | | | | |  |  |

**CONTENTS**

1. Syllabus
2. Lab Cycles
3. Instructions to Students
4. Lab Exercises
5. Additional Lab Programs
6. Viva Questions

**INSTRUCTIONS TO STUDENTS**

1. After entering the lab, make an entry in the log book providing the specified details.
2. Observation book and Lab Record should be duly signed by the concerned faculty after the completion of each experiment, failing which marks will not be awarded. All experiments must be completed as per the schedule.
3. After the completion of lab, properly shutdown the system.

**GRADING:**

* The overall lab evaluation is for **100** marks. In that **75** marks for Semester End Examination and remaining **25** marks will be awarded based on **internal evaluation**.
* Internal Evaluation consists of:
  + Lab internal examination: There will be two internal examinations for 10 marks each. The final marks are awarded as the average of two internal examination marks.
  + Continuation evaluation is for 15 marks and distributed for each experiment as:

**A. Experimentation 5 marks**

**B. Observation 5 marks**

**C. Record 5 marks**

* The overall lab evaluation is for **100** marks. In that **75** marks for Semester End Examination and remaining **25** marks will be awarded based on **internal evaluation**.

**Lab Cycle 1**

**1. Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.**

**AIM**

Implement the data link layer framing method such as bit stuffing.

**EXPLANATION:** In character stuffing read string of characters and append the

delimiters at the start and end .If the delimiter flag appears as part of the input string, then

stuff a one more delimiter in the string.

In bit stuffing, read the string of bits and append special bit pattern

01111110 as start and end delimiter .When five consecutive 1’s appears in the input bit

string stuff a 0.

**PROGRAM**

#include<stdio.h>

#include<string.h>

main()

{

char a[20],fs[50]="",t[6],r[5];

int i,j,p=0,q=0;

clrscr();

printf("enter bit string : ");

scanf("%s",a);

strcat(fs,"01111110");

if(strlen(a)<5)

{

strcat(fs,a);

}

else

{

for(i=0;i<strlen(a)-4;i++)

{

for(j=i;j<i+5;j++)

{

t[p++]=a[j];

}

t[p]='\0';

if(strcmp(t,"11111")==0)

{

strcat(fs,"111110");

i=j-1;

}

else

{

r[0]=a[i];

r[1]='\0';

strcat(fs,r);

}

p=0;

}

for(q=i;q<strlen(a);q++)

{

t[p++]=a[q];

}

t[p]='\0';

strcat(fs,t);

}

strcat(fs,"01111110");

printf("After stuffing : %s",fs);

getch();

}

**Output**

# Enter bit string : 10101111110

After stuffing : 0111111010101111101001111110

Enter bit string : 1011111011110111110

After stuffing : 0111111010111110011110111110001111110

**AIM**

Implement the data link layer framing method such as **Character Stuffing.**

#include<stdio.h>

#include<string.h>

main()

{

char a[30],fs[50]="",t[3],sd,ed,x[3],s[3],d[3],y[3];

int i,j,p=0,q=0;

clrscr();

printf("Enter characters to be stuffed : ");

scanf("%s",a);

printf("\nEnter a character that represents starting delimiter : ");

scanf(" %c",&sd);

printf("\nEnter a character that represents ending delimiter : ");

scanf(" %c",&ed);

x[0]=s[0]=s[1]=sd;

x[1]=s[2]='\0';

y[0]=d[0]=d[1]=ed;

d[2]=y[1]='\0';

strcat(fs,x);

for(i=0;i<strlen(a);i++)

{

t[0]=a[i];

t[1]='\0';

if(t[0]==sd)

strcat(fs,s);

else

if(t[0]==ed)

strcat(fs,d);

else

strcat(fs,t);

}

strcat(fs,y);

printf("\nAfter stuffing : %s",fs);

getch();

}

**Test Case**

Enter characters to be stuffed : goodday

Enter a character that represents starting delimiter : d

Enter a character that represents ending delimiter : g

After stuffing : dggooddddayg

**2. Write a program to compute CRC code for the polynomials CRC-12, CRC-16**

**and CRC CCIP**

**AIM:** To implement on a data set of characters the three CRC polynomials CRC-12, CRC-16 and CRC CCITT.

**EXPLANATION:** This algorithm is used to detect errors in the transmitted frames. The sender will compute the checksum on the data bits and this checksum will be appended to the data bits. The receiver at the other end recomputed the checksum and if it is zero, the frame contains no errors .otherwise the frame contains the error and the sender has to retransmit the

frame again. For computing the checksum, this algorithm uses generator polynomials CRC x12 +x11+x3+x2+x1+1, CRC 16 - x16+x15+x2+1,

CRC CCITT x16+x12+x5+1.

**Program:**

#include<stdio.h>

const char \* bindiv(const char \*,const char \*);

const char \* binsub(const char \*,const char \*);

int f=0,ll=0;

main()

{

char \*a,p[13]="1100000001011",g[30],g1[30],yy[30]="",td[30],\*aa;

int l=0,i;

clrscr();

printf("enter transfered data : ");

scanf("%s",g);

printf("enter received data : ");

scanf("%s",td);

strcpy(g1,g);

strcat(g,"000000000000");

printf("\n%s ) %s (",p,g);

a=bindiv(g,p);

if(strlen(a)<12)

{

for(i=strlen(a);i<12;i++)

{

yy[l++]='0';

}

yy[l]='\0';

}

strcat(yy,a);

strcat(g1,yy);

printf("\ncrc is %s",yy);

printf("\n ------------------");

strcat(td,yy);

printf("\n\n%s ) %s (",p,td);

ll=0;

aa=bindiv(td,p);

strcpy(a,aa);

printf("\n %s",a);

printf("\n -----------------");

if(f==1)

printf("\ndata transfered correctly");

else

printf("\ndata transfered incorrectly");

getch();

}

const char \* bindiv(const char \*s,const char \*d)

{

int i,j,k=0,x=13,h,p=0,l;

char q[15]="",b[30],\*w;

for(i=0;i<strlen(s);i++)

{

if((i+x)>strlen(s))

x=(i+x)-strlen(s)+1;

for(j=i;j<(i+x);j++)

{

b[k++]=s[j];

}

b[k]='\0';

if(ll!=0)

printf("\n %s",b);

ll=1;

if(strlen(b)==12)

{

break;

}

printf("\n %s",d);

printf("\n -----------------");

w=binsub(b,d);

k=0;i=j-1;

for(l=0;l<strlen(w);l++)

{

if(w[l]=='1')

break;

}

if(l==strlen(w))

{

f=1;

return(w);

}

for(h=l;h<strlen(w);h++)

{

q[p++]=w[h];

}

q[p]='\0';

x=13-strlen(q);

strcpy(b,"");

strcat(b,q);

k=strlen(q); p=0;

}

return(b);

}

const char \* binsub(const char \*x,const char \*y)

{

int i,j=0;

char w[15]="",e[3],f[3],n[3];

e[0]='1';

e[1]='\0';

f[0]='0';

f[1]='\0';

for(i=0;i<strlen(x);i++)

{

if((x[i]=='1')&&(y[i]=='1'))

strcat(w,f);

else

if((x[i]=='0')&&(y[i]=='0'))

strcat(w,f);

else

strcat(w,e);

}

n[0]='\0';

n[1]='\0';

strcat(w,n);

return(w);

}

**AIM**

Implement on a data set of characters the CRC Polynomial CRC 16

**PROGRAM**

#include<stdio.h>

const char \* bindiv(const char \*,const char \*);

const char \* binsub(const char \*,const char \*);

int f=0,ll=0;

main()

{

char \*a,p[20]="10001000000100001",

g[30],g1[30],yy[30]="",td[30],\*aa;

int l=0,i;

clrscr();

printf("enter transfered data : ");

scanf("%s",g);

printf("enter received data : ");

scanf("%s",td);

strcpy(g1,g);

strcat(g,"0000000000000000");

printf("\n%s ) %s (",p,g);

a=bindiv(g,p);

if(strlen(a)<16)

{

for(i=strlen(a);i<16;i++)

{

yy[l++]='0';

}

yy[l]='\0';

}

strcat(yy,a);

strcat(g1,yy);

printf("\n ------------------");

printf("\ncrc is %s",yy);

strcat(td,yy);

printf("\n\n%s ) %s (",p,td);

ll=0;

aa=bindiv(td,p);

strcpy(a,aa);

printf("\n %s",a);

printf("\n -----------------");

if(f==1)

printf("\ndata transfered correctly");

else

printf("\ndata transfered incorrectly");

getch();

}

const char \* bindiv(const char \*s,const char \*d)

{

int i,j,k=0,x=17,h,p=0,l;

char q[25]="",b[30],\*w;

for(i=0;i<strlen(s);i++)

{

if((i+x)>strlen(s))

x=(i+x)-strlen(s)+1;

for(j=i;j<(i+x);j++)

{

b[k++]=s[j];

}

b[k]='\0';

if(ll!=0)

printf("\n %s",b);

ll=1;

if(strlen(b)==16)

{

break;

}

printf("\n %s",d);

printf("\n -----------------");

w=binsub(b,d);

k=0;i=j-1;

for(l=0;l<strlen(w);l++)

{

if(w[l]=='1')

break;

}

if(l==strlen(w))

{

f=1;

return(w);

}

for(h=l;h<strlen(w);h++)

{

q[p++]=w[h];

}

q[p]='\0';

x=17-strlen(q);

strcpy(b,"");

strcat(b,q);

k=strlen(q); p=0;

}

return(b);

}

const char \* binsub(const char \*x,const char \*y)

{

int i,j=0;

char w[25]="",e[3],f[3],n[3];

e[0]='1';

e[1]='\0';

f[0]='0';

f[1]='\0';

for(i=0;i<strlen(x);i++)

{

if((x[i]=='1')&&(y[i]=='1'))

strcat(w,f);

else

if((x[i]=='0')&&(y[i]=='0'))

strcat(w,f);

else

strcat(w,e);

}

n[0]='\0';

n[1]='\0';

strcat(w,n);

return(w);

}

**OUTPUT**

Enter transfered data : 11011

Enter received data : 11011

10001000000100001 ) 110110000000000000000 (

10001000000100001

-------------------------

10100000001000010

10001000000100001

--------------------------

10100000110001100

10001000000100001

--------------------------

1010001101011010

--------------------------

crc is 1010001101011010

10001000000100001 ) 110111010001101011010 (

10001000000100001

--------------------------

10101010000101001

10001000000100001

--------------------------

10001000000100001

10001000000100001

--------------------------

00000000000000000

--------------------------

Data transfered correctly

**3. Develop a simple data link layer that performs the flow control using the sliding window protocol,and loss recovery using the Go-Back-N mechanism.**

**AIM:** To implement Go-Back-N mechanism

**EXPLANATION:**

Go – Back – N ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. The frames are sequentially numbered and a finite number of frames. The maximum number of frames that can be sent depends upon the size of the sending window. If the acknowledgment of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.

**Program:**

#include<stdio.h>

int main()

{

int windowsize,sent=0,ack,i;

printf("enter window size\n");

scanf("%d",&windowsize);

while(1)

{

for( i = 0; i < windowsize; i++)

{

printf("Frame %d has been transmitted.\n",sent);

sent++;

if(sent == windowsize)

break;

}

printf("\nPlease enter the last Acknowledgement received.\n");

scanf("%d",&ack);

if(ack == windowsize)

break;

else

sent = ack;

}

return 0;

}

**output**:-

enter window size

8

Frame 0 has been transmitted.

Frame 1 has been transmitted.

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted.

Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

2

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted.

Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

8

**Lab Cycle 2**

**4. Implement Dijsktra’s algorithm to compute the shortest path through a network**

**AIM**

Implement Dijkstra’s algorithm to compute the Shortest path through a graph.

**EXPLANTION:**

Build a graph of the subnet, with each node representing a router and each arc of the graph of a communication link. The labels on the arc could be computed as function of delay. Initially, no paths are known so all the nodes are labeled to infinity ,as the algorithm proceeds and found labels may change reflecting better paths .Initially all labels are tentative when it is discovered that a label represent shortest path from the source node o that node it is made permanent and never changed thereafter. Each node is labeled with its distance from the source node along the best known path. The distance will be the shortest from source to destination.

**PROGRAM**

#include<stdio.h>

void sort(void);

static int dsp[10][10],nodes;

struct{

char src;

char dest;

int length;

}stemp,permanent[10]={' ',' ',0},temp[10]={' ',' ',-1};

static int perm,tem;

void main()

{

int i,j,k,l,m,n=0,point;

char initial,dest,path[10]={' '};

clrscr();

printf("\t\t Shortest Path (Dijkstra's algorithm)");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf(“\nEnter the number of nodes:”);

scanf(“%d”,&nodes);

printf(“\nEnter the adjacency matrix for the graph:\n”);

for(i=0;i<nodes;i++)

{

for(j=0;j<nodes;j++)

scanf(“%d”,&dsp[I][j]);

}

fflush(stdin);

printf("\n enter the source node:");

scanf("%c",&initial);fflush(stdin);

printf("\n Enter the destination node:");

scanf("%c",&dest);

permanent[perm].src=initial;

permanent[perm].dest=initial;

permanent[perm++].length=0;

i=permanent[perm-1].dest-97;

for(j=0;j<nodes;j++)

{

if(i!=j)

{

if(dsp[i][j]>0)

{

temp[tem].src=permanent[perm-1].src;

temp[tem].dest=j+97;

temp[tem++].length=dsp[i][j];

}

}

}

sort();

while(tem>=0)

{

j=permanent[perm-1].dest-97;

for(i=0;i<nodes;i++)

{

if(i!=initial-97)

{

if(dsp[j][i]>0)

{

l=-1;

for(k=0;k<perm;k++)

{

if(permanent[k].dest==(i+97))

l=k;

}

for(k=0;k<=tem;k++)

{

if(temp[k].dest==(i+97))

l=k;

}

if(l<0)

{

temp[tem].src=j+97;

temp[tem].dest=i+97;

for(m=0;m<perm;m++)

{

if(permanent[m].dest==temp[tem].src)

n=permanent[m].length;

}

temp[tem++].length=dsp[j][i]+n;

}

else

{

for(m=0;m<perm;m++)

{

if(permanent[m].dest==j+97)

{

n=permanent[m].length+dsp[j][i];break;

}

else

n=dsp[j][i];

}

if((n<temp[l].length))

{

temp[l].length=n;

temp[l].src=j+97;

temp[l].dest=i+97;

}

}

}

}

}

sort();

}

printf("\nShortest path:\n");

printf("From %c to %c is:",initial,dest);

for(i=0;i<perm-1;i++)

{

if(permanent[i].dest==dest)

{

point=i;n=i; break;

}

} i=0;

for(j=perm;j>0;j--)

{

if(permanent[j-1].dest==permanent[point].src)

{

path[i++]=permanent[point].dest;

point=j-1;

}

}

path[i]=initial;

for(j=i;j>=0;j--)

printf("%c ",path[j]);

printf("\t length=%d",permanent[n].length);

getch();

}

void sort()

{

int i,j,k;

for(i=0;i<=tem;i++)

{

k=1;

for(j=0;j<=tem;j++)

{

if((temp[j].length <= temp[j+1].length))

{

stemp=temp[j];

temp[j]=temp[j+1];

temp[j+1]=stemp; k=0;

}

}

if(k)

break;

}

permanent[perm++]=temp[tem-1];

temp[tem-1].src=' ';temp[tem-1].dest=' ';

temp[tem-1].length=-1; tem--;

}

**Network topology:**

**------------------------ 1**

**1 b c 1**

**a 1 e 3 f**

**2 2**

**d**

**Output of execution1:**

**----------------------------**

**Shortest Path (Dijkstra's algorithm)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** Enter the number of nodes:6

Enter the adjacency matrix for the graph(-1 for no edge):

0 1 -1 2 -1 -1

1 0 1 -1 -1 -1

-1 1 0 1 1 -1

2 -1 1 0 2 -1

-1 -1 1 2 0 3

-1 -1 -1 -1 3 0

Enter the source node:a

Enter the destination node:f

Shortest path:

From a to f is:a b c e f length=6

**Network Topology:**

**B 7 C**

**2 2 3 3**

**A E 2 F D**

**6 1 2 2**

**G 4 H**

**Output of execution2:**

**----------------------------**

**Shortest Path (Dijkstra's algorithm)**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Enter the number of nodes:8

Enter the adjacency matrix for the graph(-1 for no edge):

Enter the adjacency matrix for the graph

0 2 -1 -1 -1 -1 6 -1

2 0 7 -1 2 -1 1 -1

-1 7 0 3 -1 3 -1 -1

-1 -1 3 0 -1 -1 -1 2

-1 2 -1 -1 0 2 1 -1

-1 -1 3 -1 2 0 -1 2

6 -1 -1 -1 1 -1 0 4

-1 -1 -1 2 -1 2 4 0

Enter the source node:a

Enter the destination node:d

Shortest path:

From a to d is:a b e f h d length=10

**5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.**

**AIM:**

Take an example subnet graph with weights indicating delay between nodes.

Now obtain Routing table art each node using distance vector routing algorithm.

**EXPLANATION:**

Sending a packet to all the destinations simultaneously is called as broadcasting. In a subnet when a sender host wants to broadcasts packets, it has to construct a spanning tree and then only its sends packets to all other hosts. This method will avoid duplicate packets in the subnet.

**PROGRAM**

#include<stdio.h>

int max();

int distance[20];

int n;

main()

{

int adj[20][20],adj1[20][20],flag[30];

int i,j,root,x;

int source,count=1,y=0;

printf("enter no of nodes");

scanf("%d",&n);

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

scanf("%d",&adj[i][j]);

}

}

printf("enter the source for broadcasting");

scanf("%d",&source);

for(i=0;i<n;i++)

{

flag[i]=0;

}

for(root=0;root<n;root++)

{

for(i=0;i<n;i++)

{

distance[i]=adj[root][i];

}

x=min();

for(i=0;i<n;i++)

{

if(distance[i]==x)

{

adj1[root][i]=x;

adj1[i][root]=x;

}

else

{

adj1[root][i]=0;

}

}

}

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

if(adj1[i][j]!=0)

{

adj1[j][i]=adj[i][j];

}

}

}

printf("given adjacency matrix is");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d",adj[i][j]);

}

printf("\n");

}

printf("minimal spanning tree");

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

printf("%d ",adj1[i][j]);

}

printf("\n");

}

root=source;

flag[root]=1;

while(count!=y)

{

for(i=0;i<n;i++)

{

if(adj1[root][i]!=0 && flag[root]==1 && flag[i]!=1)

{

printf("%d sends message to %d \n",root,i);

flag[i]=1;

}

}

if(root<n-1)

{

root++;

}

else

{

root=0;

}

for(i=0;i<n;i++)

{

if(flag[i]==0)

{

break;

}

}

if(i==n)

{

count=y;

}

}

}

int min()

{

int i,j=0;

int mini;

int distance1[10];

for(i=0;i<n;i++)

{

if(distance[i]!=0)

{

distance1[j]=distance[i];

j++;

} }

mini=distance1[0];

for(i=1;i<j;i++)

{

if(distance1[i]<mini)

{

mini=distance1[i];

}

} return(mini);

}

**OUTPUT**

enter no of nodes2

enter the adjacency matrix

0 2

2 0

enter the source for broadcasting1

given adjacency matrix is

0 2

2 0

2

minimal spanning tree is

0 2

2 0

1 sends message to 0

2

enter no of nodes3

enter the adjacency matrix

0 1 2

1 0 5

2 5 0

enter the source for broadcasting2

given adjacency matrix is

0 1 2

1 0 5

2 5 0

1

2 5

minimal spanning tree is

0 1 2

1 0 0

2 0 0

1

2

2 sends message to 0

0 sends message to 1

enter no of nodes4

enter the adjacency matrix

0 6 8 7

6 0 5 0

8 5 0 4

7 0 4 0

enter the source for broadcasting2

given adjacency matrix is

0 6 8 7

6 0 5 0

8 5 0 4

7 0 4 0

4

7 8 5

6

minimal spanning tree is

0 6 0 0

6 0 5 0

0 5 0 4

0 0 4 0

4

5

6

2 sends message to 1

2 sends message to 3

1 sends message to 0

enter no of nodes5

enter the adjacency matrix

0 2 0 5 0

2 0 3 6 0

0 3 0 7 9

5 6 7 0 8

0 0 9 8 0

enter the source for broadcasting2

given adjacency matrix is

0 2 0 5 0

2 0 3 6 0

0 3 0 7 9

5 6 7 0 8

0 0 9 8 0

3

2

6 7 9

5

8

minimal spanning tree is

0 2 0 5 0

2 0 3 0 0

0 3 0 0 0

5 0 0 0 8

0 0 0 8 0

3

2

5

8

2 sends message to 1

1 sends message to 0

0 sends message to 3

3 sends message to 4

enter the no of nodes6

enter the adjacency matrix

0 2 0 7 6 0

2 0 3 0 0 0

0 3 0 4 0 0

7 0 4 0 5 0

6 0 0 5 0 7

0 0 0 0 7 0

enter the source for broadcasting3

given adjacency matrix is

0 2 0 7 6 0

2 0 3 0 0 0

0 3 0 4 0 0

7 0 4 0 5 0

6 0 0 5 0 7

0 0 0 0 7 0

6 7

2 8 5

3 4

minimal spanning tree is

0 2 0 0 0 0

2 0 3 0 0 0

0 3 0 4 0 0

0 0 4 0 5 0

0 0 0 5 0 7

0 0 0 0 7 0

7

2 5

3 4

3 sends message to 2

3 sends message to 4

4 sends message to 5

2 sends message to 1

1 sends message to 0

**6. Implement distance vector routing algorithm for obtaining routing tables at each**

**node.**

**Program**

#include &lt;conio.h&gt;

#include &lt;iostream.h&gt;

#define MAX 10

int n;

class router {

char adj\_new[MAX],

adj\_old[MAX];

int table\_new[MAX], table\_old[MAX];

public:

router(){

for(int i=0;i&lt;MAX;i++) table\_old[i]=table\_new[i]=99;

}

void copy(){

for(int i=0;i&lt;n;i++) {

adj\_old[i] =adj\_new[i];

table\_old[i]=table\_new[i];

}

}

int equal() {

for(int i=0;i&lt;n;i++)

if(table\_old[i]!=table\_new[i]||adj\_new[i]!=adj\_old[i])return 0;

return 1;

}

void input(int j) {

printf(&quot;Enter 1 if the corresponding router is adjacent to router (char(&#39;A&#39;+j) else enter 99: &quot;);

for(int i=0;i&lt;n;i++)

if(i!=j) printf(“%c”,(char(&#39;A&#39;+i)));

printf(&quot;\nEnter matrix:&quot;);

for(i=0;i&lt;n;i++) {

if(i==j)

table\_new[i]=0;

else

scanf(“%d”,table\_new[i]);

adj\_new[i]= (char)(&#39;A&#39;+i);

}

}

void display(){

printf(&quot;\nDestination Router: &quot;);

for(int i=0;i&lt;n;i++)

printf(“%c”,(char)(&#39;A&#39;+i));

printf(&quot;\nOutgoing Line:”);

for(i=0;i&lt;n;i++)

printf(“%d”,adj\_new[i]);

printf(&quot;\nHop Count:”);

for(i=0;i&lt;n;i++)

printf(“%d”,table\_new[i]&quot; &quot;);

}

void build(int j) {

for(int i=0;i&lt;n;i++)

for(int k=0;(i!=j)&amp;&amp;(k&lt;n);k++)

if(table\_old[i]!=99)

if((table\_new[i]+r[i].table\_new[k])&lt;table\_new[k]) {

table\_new[k]=table\_new[i]+r[i].table\_new[k];

adj\_new[k]=(char)(&#39;A&#39;+i);

}

}

} r[10];

void build\_table() {

int i=0, j=0;

while(i!=n) {

for(i=j;i&lt;n;i++) {

r[i].copy();

r[i].build(i);

}

for(i=0;i&lt;n;i++)

if(!r[i].equal()) {

j=i;

break;

}

}

}

void main() {

clrscr();

printf(&quot;Enter the number the routers &quot;);

scanf(“%d”,n);

for(int i=0;i&lt;n;i++) r[i].input(i);

build\_table();

for(i=0;i&lt;n;i++) {

printf(&quot;Router Table entries for router %c&quot;,(char)(&#39;A&#39;+i));

r[i].display();

}

}

**Output**

Enter the number the routers: 5

Enter 1 if the corresponding is adjacent to router A else enter 99:

B C D E

Enter matrix:1 1 99 99

Enter 1 if the corresponding is adjacent to router C else enter 99:

A C D E

Enter matrix:1 99 99 99

Enter 1 if the corresponding is adjacent to router B else enter 99:

A B D E

Enter matrix:1 99 1 1

Enter 1 if the corresponding is adjacent to router D else enter 99:

A B C E

Enter matrix:99 99 1 99

Enter 1 if the corresponding is adjacent to router E else enter 99:

A B C D

Enter matrix:99 99 1 99

Router Table entries for router

Destination Router: A B C D E

Outgoing Line:

A B C C C

Hop Count:

0 1 1 2 2

Router Table entries for router

Destination Router: A B C D E

Outgoing Line:

A B A A A

Hop Count:

1 0 2 3 3

Router Table entries for router

Destination Router: A B C D E

Outgoing Line:

A A C D E

Hop Count:

1 2 0 1 1

Router Table entries for router

Destination Router: A B C D E

Outgoing Line:

C C C D C

Hop Count:

2 3 1 0 2

Router Table entries for router

Destination Router: A B C D E

Outgoing Line:

C C C C E

Hop Count:

2 3 1 2 0

**Lab Cycle 3**

**7. Implement data encryption and data decryption(RSA)**

**AIM:** To implement RSA algorithm.

**EXPALANTION:** This a public key algorithm devised by Ron Rivest ,Shamir and Adleman. This algorithm uses two public keys namely public key and private key. Given plaintext will be encrypted using public key and the encrypted text will be converted back to plaintext using the private key.

**PROGRAM**

/\*\*\*\*\*\*\*\*\*\* RSA PROGRAM \*\*\*\*\*\*\*\*\*\*\*/

/\*d value should be less than 11 bcoz (c^d)modn can't be computed using available datatypes\*/

#include<stdio.h>

#include<string.h>

#include<math.h>

void main()

{

char a[]={"0ABCDEFGHIJKLMNOPQRSTUVWXYZ"};

int n,i,j,s,n2,k1,p,q,d,m1,e1,l5,z,p2[30],s1,c[30];

unsigned long int l3,m,l4,k2;

double l2,l1,l6;

float e,l;

char p1[30];

clrscr();

printf("enter two prime numbers p and q\n");

scanf("%d %d",&p,&q);

do{

n=p\*q;

if(n<26)

{ printf("\n n value is not large enough.\nplease select p, q value such that p\*q is greater than 26");

scanf("%d %d",&p,&q);}

}while(n<26);

z=((p-1)\*(q-1));

printf("enter the value of d:\n");

scanf("%d",&d);

for(j=1;j<z;j++)

{

if((j\*d)%z==1)

break;

}

e=j;

printf("%d %d %f\n",n,z,e);

printf("ENCRYPTION-CIPHERTEXT");

printf("enter the plain text\n");

scanf("%s",p1);

for(i=0;i<strlen(p1);i++)

{

for(j=1;j<strlen(a);j++)

{

if(a[j]==p1[i])

{

s=j;

break;

}

else

continue;

}

printf("%d",s);

e1=(int)e;

l1=pow(((double)s),((double)e1));

k2=fmod(l1,(double)n);

printf("\n%lu\n",k2);

c[i]=(int)k2;

printf("cipher:%d\n",c[i]);

}

printf("\n");

for(i=0;i<strlen(p1);i++)

{

l2=(pow(((double)c[i]),((double)d)));

m=fmod(l2,(double)n);

m1=(int)m;

printf(" %c\n",a[m1]);}

getch();

}

**OUTPUT**

enter two prime numbers p and q

3 11

enter the value of d:

7

33 20 3.000000

ENCRYPTION-CIPHERTEXTenter the plain text

SUZANNE

19

28

cipher:28

21

21

cipher:21

26

20

cipher:20

1

1

cipher:1

14

5

cipher:5

14

5

cipher:5

5

26

cipher:26

S

U

Z

A

N

N

E

RSA output:

enter two prime numbers p and q

5 13

enter the value of d:

7

65 48 7.000000

ENCRYPTION-CIPHERTEXTenter the plain text

NAINA

14

14

cipher:14

1

1

cipher:1

9

9

cipher:9

14

14

cipher:14

1

1

cipher:1

N

A

I

N

A

**8. Write a program for congestion control using Leaky bucket algorithm.**

**AIM**: Implement Leaky Bucket algorithm.

**Explanation:** Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.

**Program**

#include<stdio.h>

#include<stdlib.h>

#define bucketSize 512

void bktInput(int a,int b)

{

if(a>bucketSize)

printf("\n\t\tBucket overflow");

else {

delay(500);

while(a>b){

printf("\n\t\t bytes outputed.");

a-=b;

delay(500);

}

if (a>0) printf("\n\t\tLast %d%d" ,&a,&b);

printf(“bytes sent\t");

printf("\n\t\tBucket output successful");

}

}

void main() {

int op, pktSize;

randomize();

printf("Enter output rate : ");

scanf(“%d”,&op);

for(int i=1;i<=5;i++){

delay(random(1000));

pktSize=random(1000);

printf("\nPacket no %d",i);

printf("\tPacket size = %d",pktSize);

bktInput(pktSize,op);

}

}

**Output**

Enter output rate : 100

Packet no 0 Packet size = 3

Bucket output successful

Last 3 bytes sent

Packet no 1 Packet size = 33

Bucket output successful

Last 33 bytes sent

Packet no 2 Packet size = 117

Bucket output successful

100 bytes outputted.

Last 17 bytes sent

Packet no 3 Packet size = 95

Bucket output successful

Last 95 bytes sent

Packet no 4 Packet size = 949

Bucket overflow

**9. Write a program for frame sorting technique used in buffers.**

**AIM:**

program for frame sorting technique used in buffers.

**EXPLANATION:**

The data link layer divides the stream of bits received from the network layer into manageable

data units called frames.

If frames are to be distributed to different systems on the network, the Data link layer adds a

header to the frame to define the sender and/or receiver of the frame.

Each Data link layer has its own frame format. One of the fields defined in the format is the

maximum size of the data field. In other words, when datagram is encapsulated in a frame, the

total size of the datagram must be less than this maximum size, which is defined by restriction

imposed by the hardware and software used in the network.

Program

#include<stdlib.h>

#include<time.h>

#include<stdio.h>

#include<conio.h>

#include<string.h>

#define FSize 3

typedef struct packet{int SeqNum; char Data[FSize+1];}packet;

struct packet \*readdata, \*transdata;

int divide(char \*msg) {

int msglen, NoOfPacket, i, j;

msglen = strlen(msg);

NoOfPacket = msglen/FSize;

if ((msglen%FSize)!=0) NoOfPacket++;

readdata = (struct packet \*)malloc(sizeof(packet) \* NoOfPacket);

for(i = 0; i < NoOfPacket; i++)

{

readdata[i].SeqNum = i + 1;

for (j = 0; (j < FSize) && (\*msg != '\0'); j++, msg++)

readdata[i].Data[j] = \*msg;

readdata[i].Data[j] = '\0';

}

printf("\nThe Message has been divided as follows\n");

printf("\nPacket No.Data\n\n");

for (i = 0; i < NoOfPacket; i++)

printf("%2d %s\n", readdata[i].SeqNum,readdata[i].Data);

return NoOfPacket;

}

void shuffle(int NoOfPacket)

{

int \*Status;

int i, j, trans;

randomize();

Status=(int \* )calloc(NoOfPacket, sizeof(int));

transdata = (struct packet \*)malloc(sizeof(packet) \* NoOfPacket);

for (i = 0; i < NoOfPacket;) {

trans = rand()%NoOfPacket;

if (Status[trans]!=1) {

transdata[i].SeqNum = readdata[trans].SeqNum;

strcpy(transdata[i].Data, readdata[trans].Data);

i++;

Status[trans] = 1;

}

}

free(Status);

}

void sortframes(int NoOfPacket) {

packet temp;

int i, j;

for (i = 0; i < NoOfPacket; i++)

for (j = 0; j < NoOfPacket – i-1; j++)

if (transdata[j].SeqNum > transdata[j + 1].SeqNum) {

temp.SeqNum = transdata[j].SeqNum;

strcpy(temp.Data, transdata[j].Data);

transdata[j].SeqNum = transdata[j + 1].SeqNum;

strcpy(transdata[j].Data, transdata[j + 1].Data);

transdata[j + 1].SeqNum = temp.SeqNum;

strcpy(transdata[j + 1].Data, temp.Data);

}

}

void receive(int NoOfPacket)

{

int i;

printf("\nPackets received in the following order\n");

for (i = 0; i < NoOfPacket; i++) printf("%4d", transdata[i].SeqNum);

sortframes(NoOfPacket);

printf("\n\nPackets in order after sorting..\n");

for (i = 0; i < NoOfPacket; i++) printf("%4d", transdata[i].SeqNum);

printf("\n\nMessage received is :\n");

for (i = 0; i < NoOfPacket; i++) printf("%s", transdata[i].Data);DBIT

Computer Networks Lab Manual

}

void main()

{

char \*msg;

int NoOfPacket;

clrscr();

printf("\nEnter The message to be Transmitted :\n");

scanf("%[^\n]", msg);

NoOfPacket = divide(msg);

shuffle(NoOfPacket);

receive(NoOfPacket);

free(readdata);

free(transdata);

getch();

}

Output

Enter The messgae to be Transmitted :

hi, it was nice meeting u on sunday

The Message has been divided as follows

Packet No.

Data

1

2

3

4

5

6

7

8

9

10

11

12

hi,

it

wa

s n

ice

me

eti

ng

u o

n s

und

ay

Packets received in the following order

4

2

6

3

5

1

8

9 11

7 12 10

Packets in order after sorting..

1

2

3

4

5

6

7

8 11 12

9

10

Message received is :

hi, it was nice meeting u on sunday

**Lab Cycle 4**

**10. Wireshark**

Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.

**i) Starting** **wireshark:**

Two different methods for starting Wireshark are available. These include the Start menu and the Run command box.

**Method 1 - Start Menu:**To start Wireshark using the Start menu:

1. Open the **Start** menu.
2. Select **All Programs**.
3. Select **Wireshark**.

**Method 2 - Run Command**:

To start Wireshark using the Run command box:

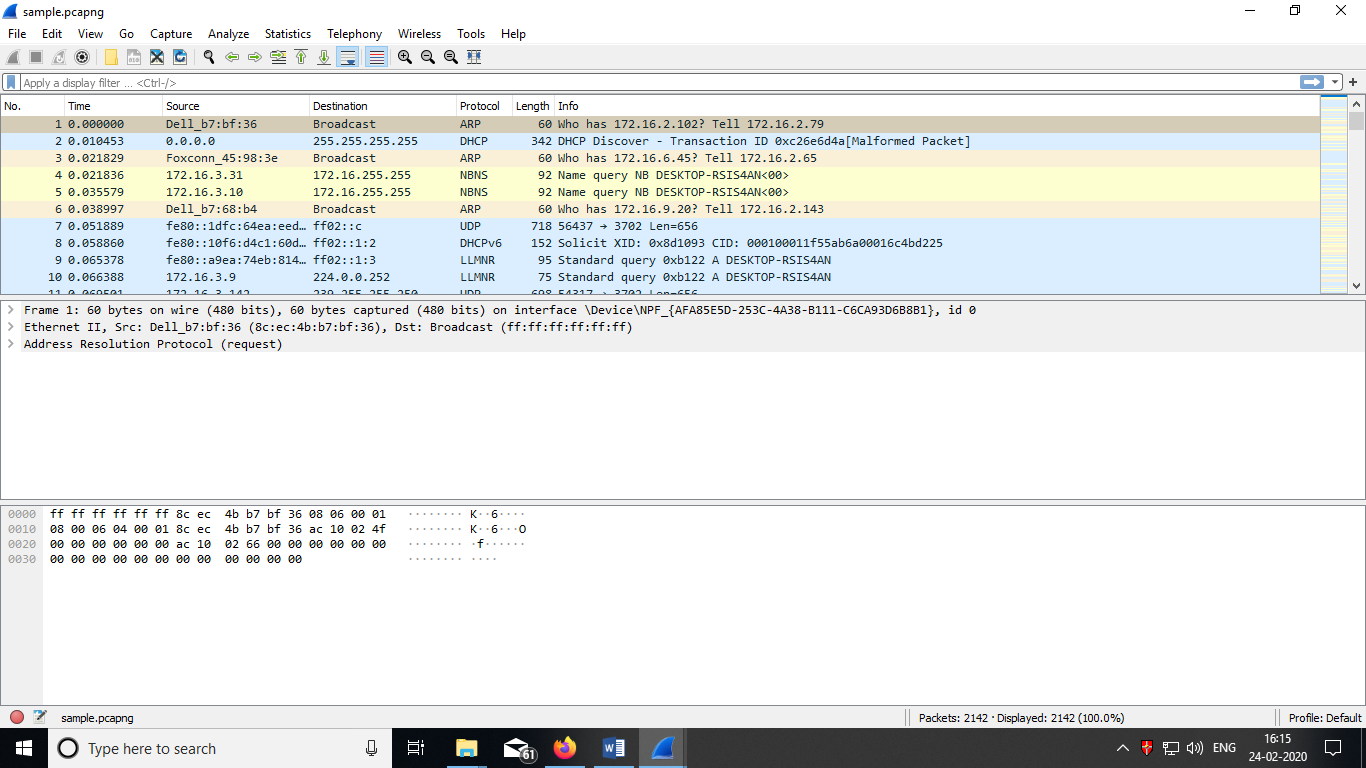
1. Open the **Start** menu or press the **Windows key + R**.
2. Type **Wireshark** in the Run command box.
3. Press **Enter**.

**ii) Packet Capture using Wireshark:**

The following methods can be used to start capturing packets with Wireshark:

* You can double-click on an interface in the [welcome screen](https://www.wireshark.org/docs/wsug_html_chunked/ChCapInterfaceSection.html).
* You can select an interface in the [welcome screen](https://www.wireshark.org/docs/wsug_html_chunked/ChCapInterfaceSection.html), then select Capture → Start or click the first toolbar button.
* You can get more detailed information about available interfaces using  [“The “Capture Options” Dialog Box”](https://www.wireshark.org/docs/wsug_html_chunked/ChCapCaptureOptions.html) (Capture → Options…).
* After some time interval we need to stop the capturing .(click stop from capture menu).

**Sample Packet Capturing (snapshot)**



**iii) Saving the Capture**

1. To save the capture, click **File** > **Save**.

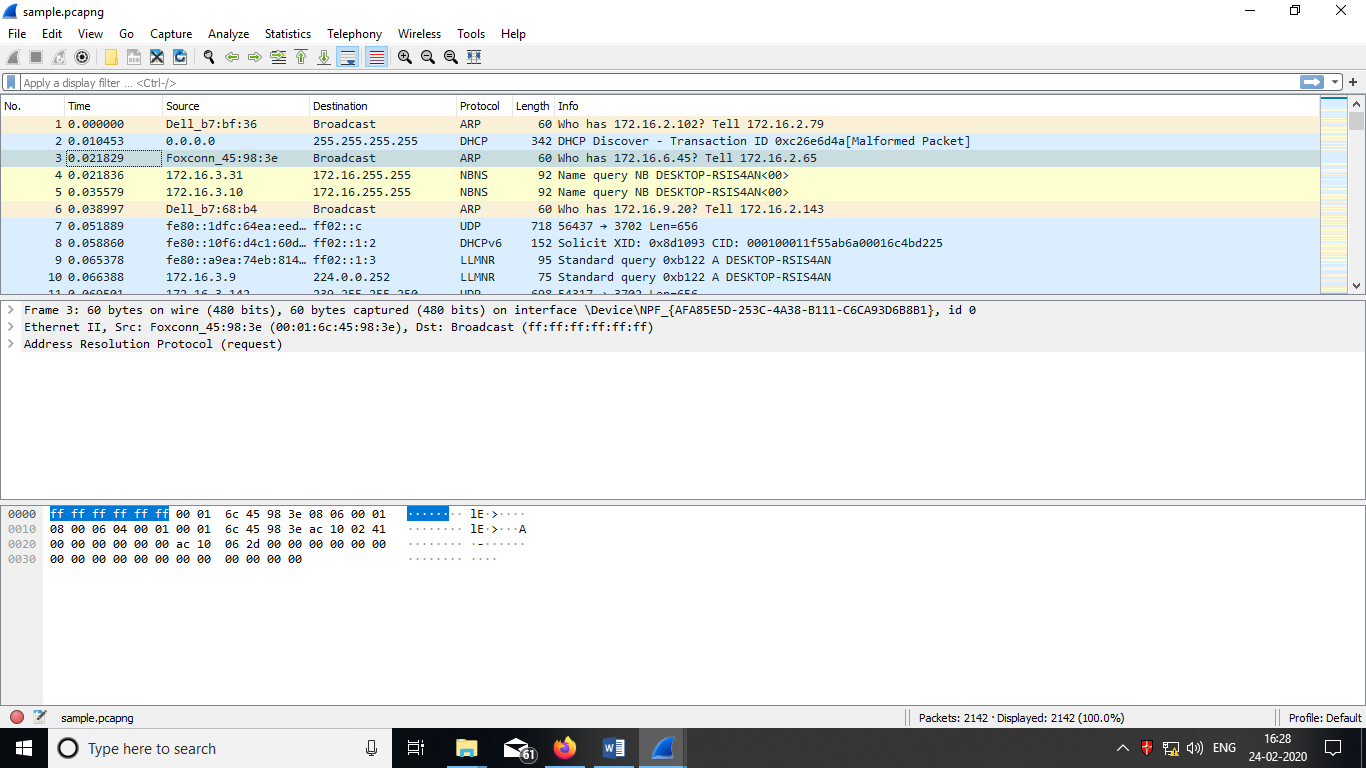
2. Name the file, and click **Save**.

*Note:* ***.Pcap*** *and.****Pcap-ng****are good file types to use for the capture if you plan to use Eye P.A. to open the capture.*

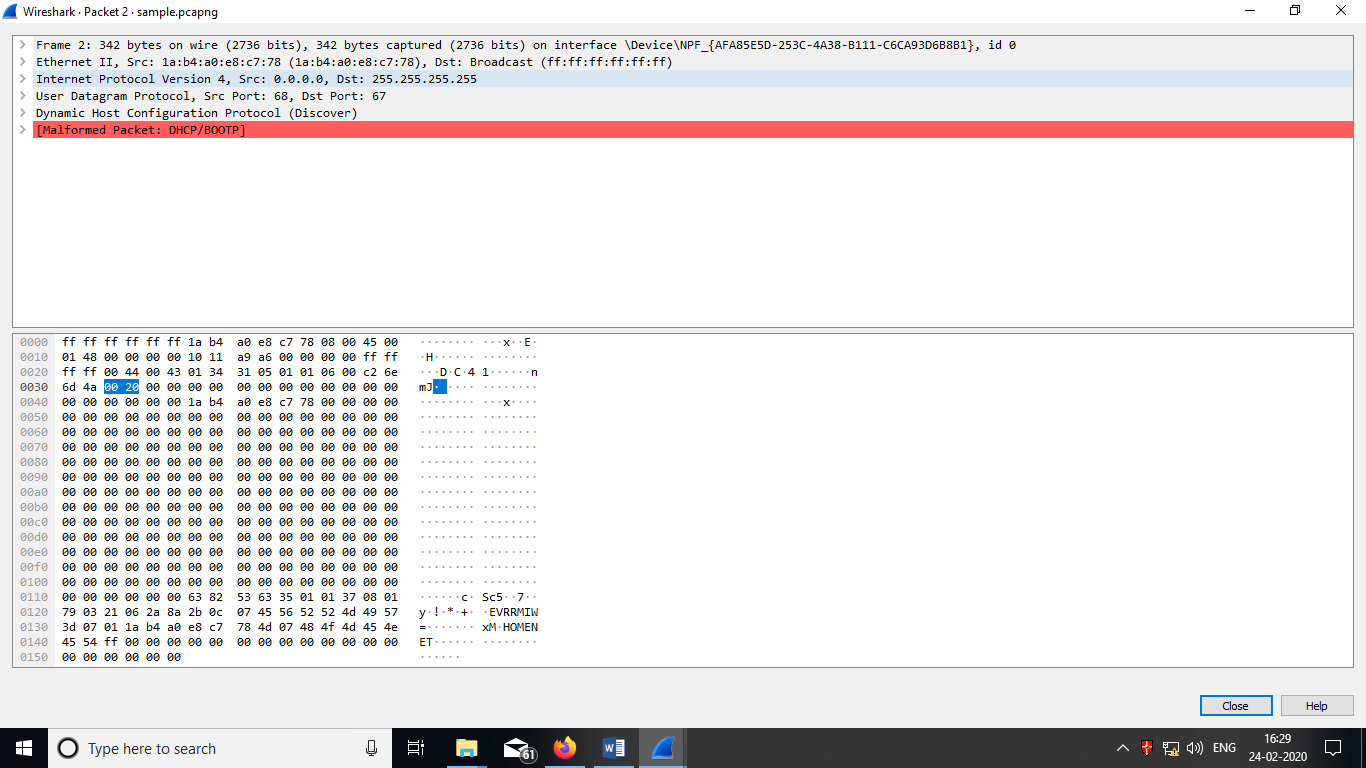
## VIEWING/ANALYSIS OF PACKETS CAPTURED

Once you have captured some packets or you have opened a previously saved capture file, you can view the packets that are displayed in the **packet list pane** (first pane) by simply clicking on a packet in the packet list pane, which will bring up the selected packet in the **packet list details** view and byte view panes.

You can then expand any part of the tree to view detailed information about each protocol in each packet. Clicking on an item in the tree will highlight the corresponding bytes in the **byte view pane.**



In addition you can view individual packets in a separate window. We can do this by double-clicking on an item in the packet list or by selecting the packet in which you are interested in the packet list pane and selecting View → Show Packet in New Window. This allows you to easily compare two or more packets, even across multiple files.



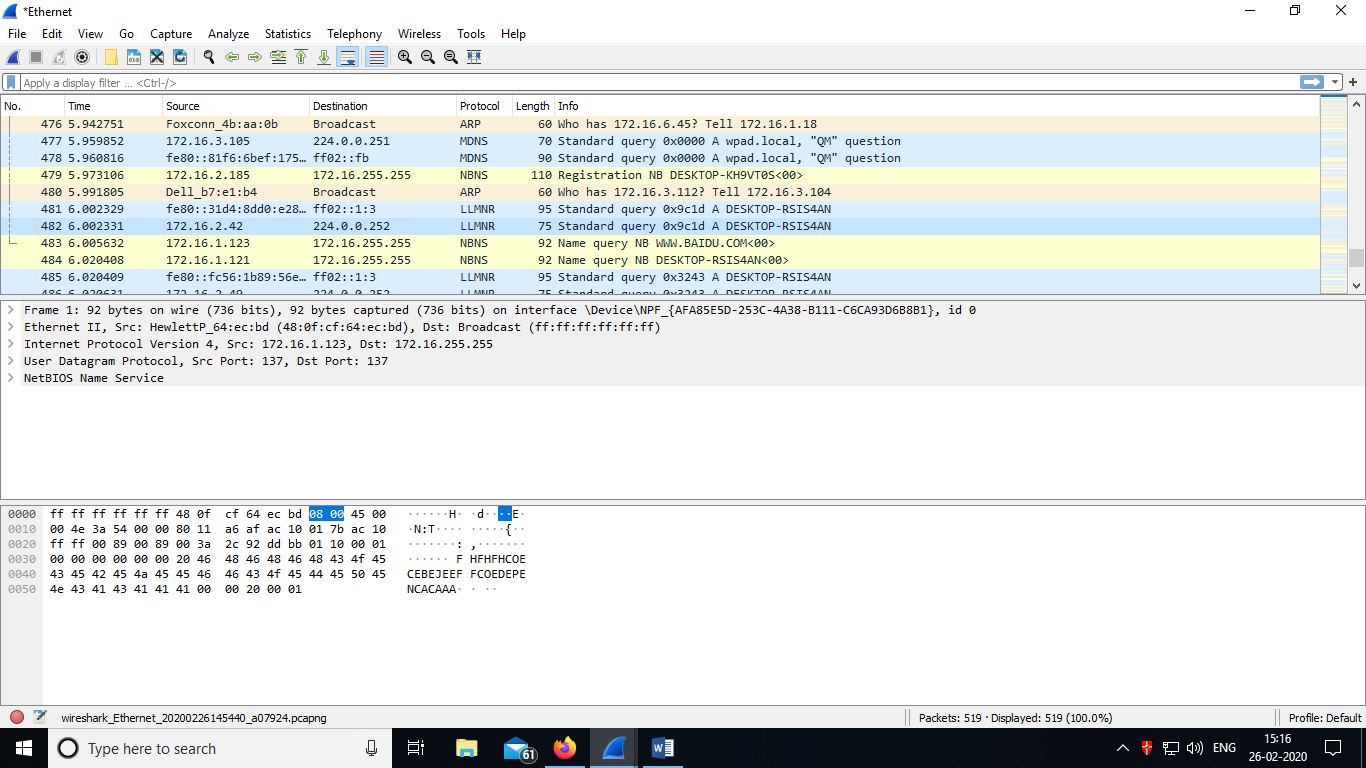
**iv) Statistics :**

When using Wireshark, we have various types of Statistic tools, starting from the simple tools for listing end-nodes and conversations, to the more sophisticated tools such as flow and I/O graphs. Here we are representing two of them.

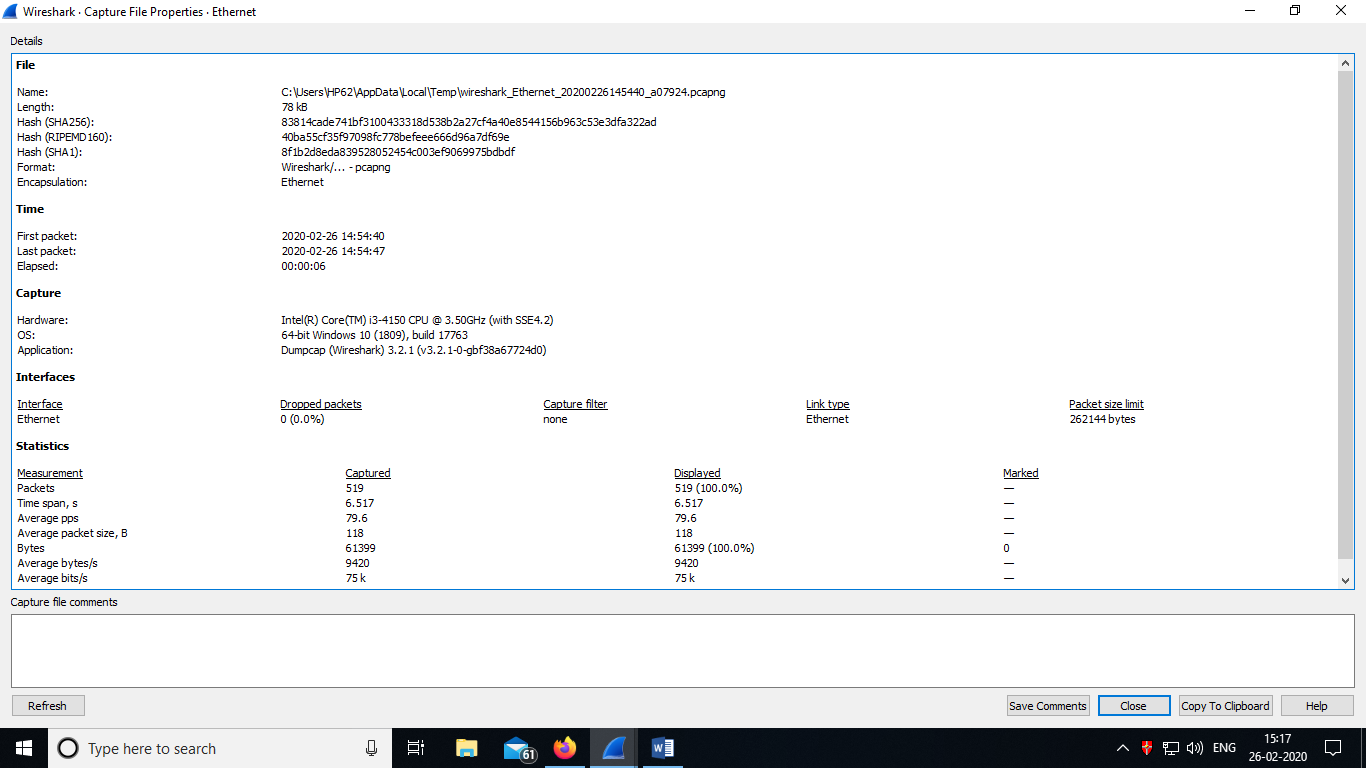
To start statistics tools, start Wireshark, and choose Statistics from the main menu.

1. **Using the statistics to capture file properties**

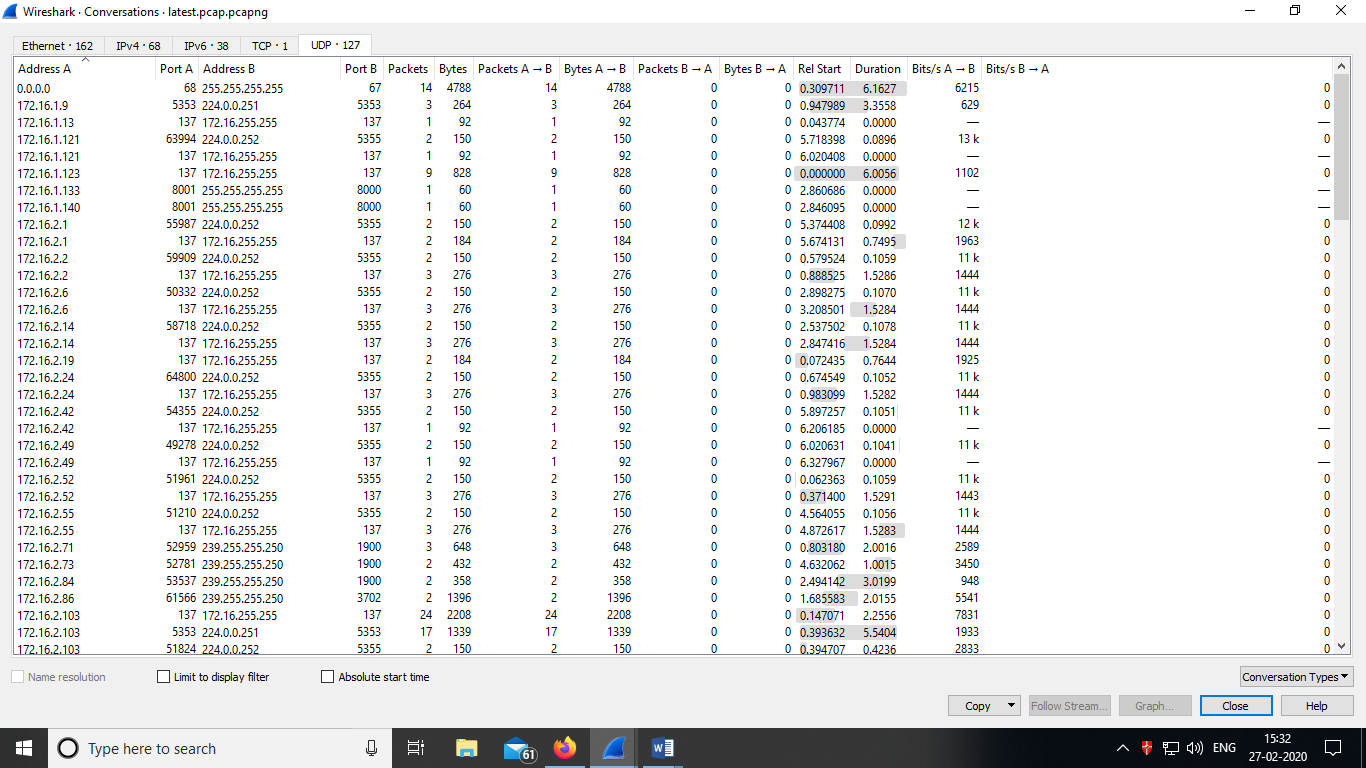
Two steps to be performed:1) Open/Create Packet Capture File.



**2) From the Statistics menu, choose Capture File Properties**



## USING THE STATISTICS TO CAPTURE CONVERSATIONS



We can use this statistics tools for:

* **On layer 2 (Ethernet)**: To find and isolate broadcast storms
* **On layer 3/layer 4 (TCP/IP)**: To connect in parallel to the internet router port, and check who is loading the line to the ISP

**V) Filters:**

Wireshark has two filtering languages: ***capture filters*** and ***display filters***. *Capture filters* are used for filtering when capturing packets.

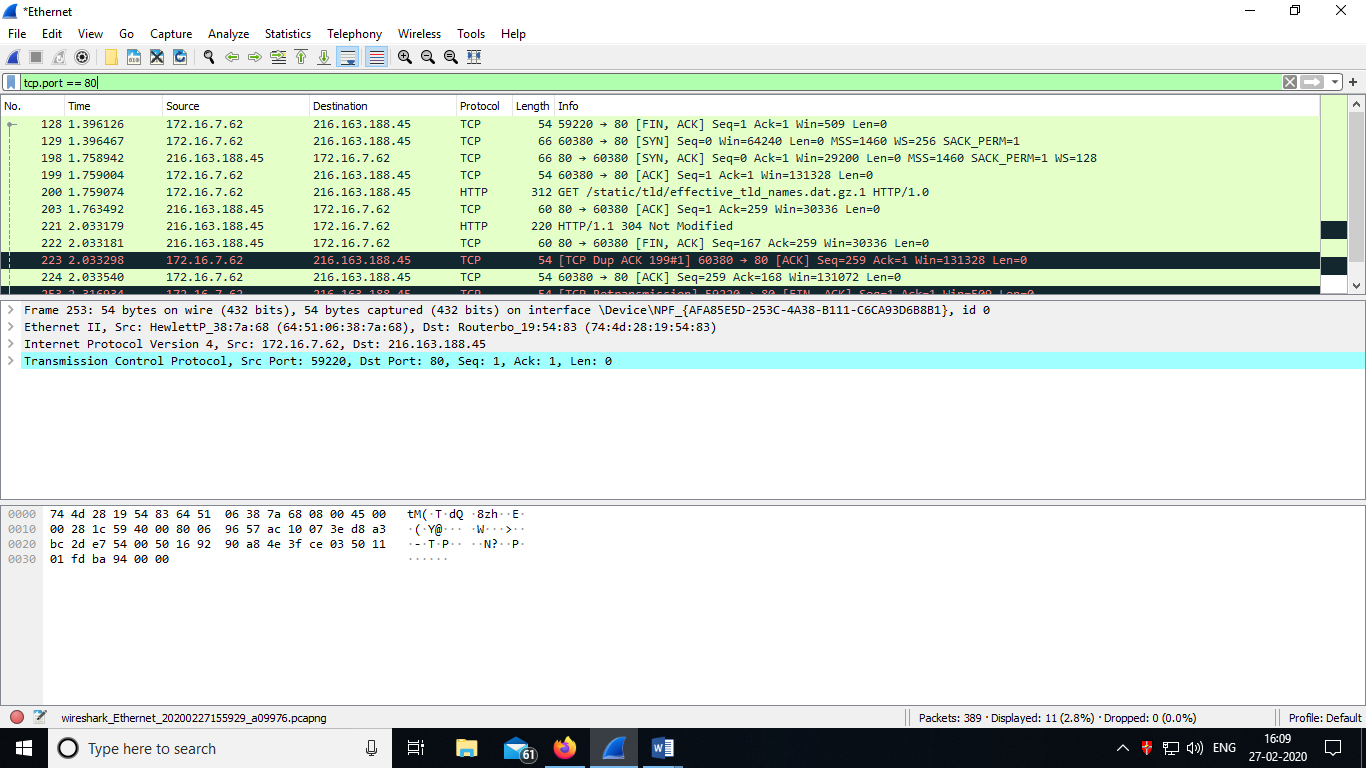
You enter the capture filter into the “Filter” field of the Wireshark “Capture Options” dialog box, from capture menu.

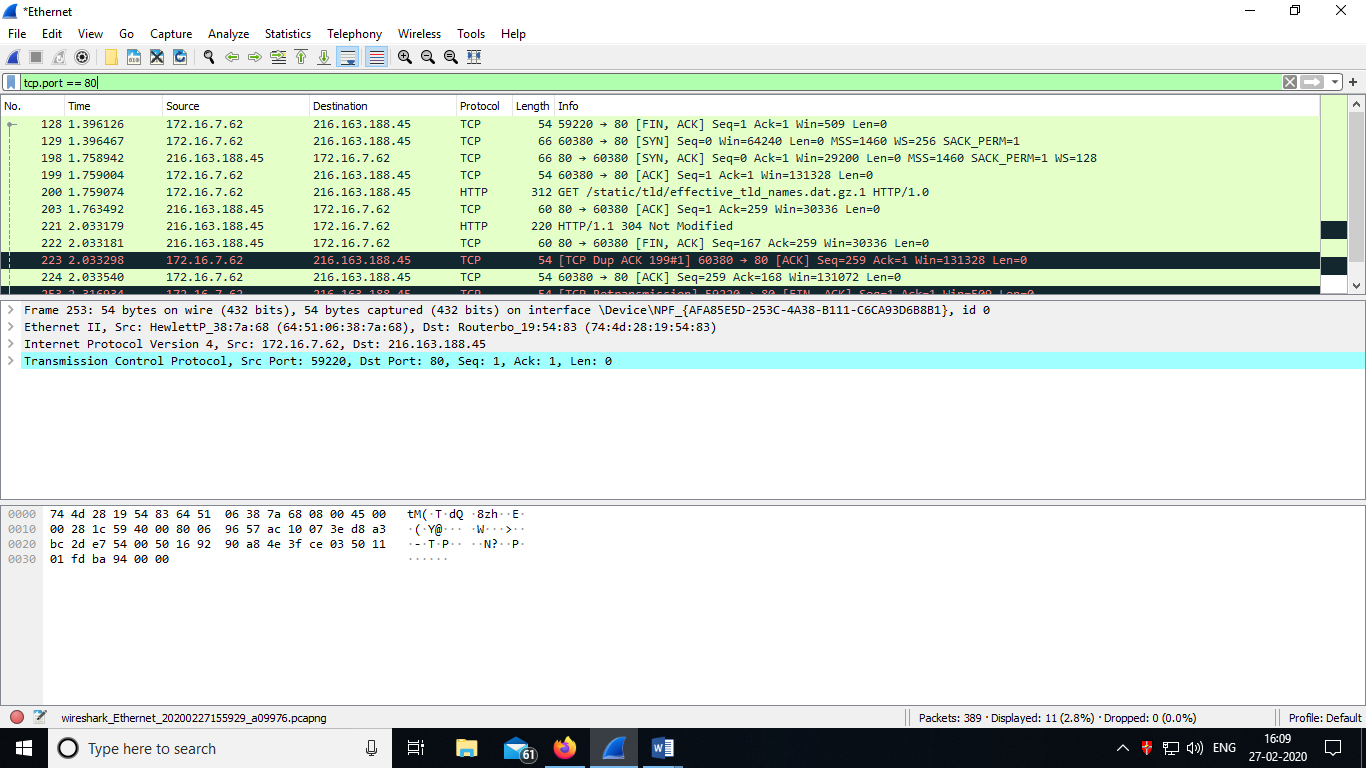
*Display filters* are used for filtering which packets are displayed .Display filters allow you to concentrate on the packets you are interested in while hiding the currently uninteresting ones. They allow you to only display packets based on:

* Protocol
* The presence of a field
* The values of fields
* A comparison between fields
* …​ and a lot more!

To only display packets containing a particular protocol, type the protocol name in the **display filter toolbar** of the Wireshark window and press enter to apply the filter.

For example if we select tcp protocol in Display filter, the following screen will be displayed.



To remove the filter, click on the Clear button to the right of the display filter field. All packets will become visible again.

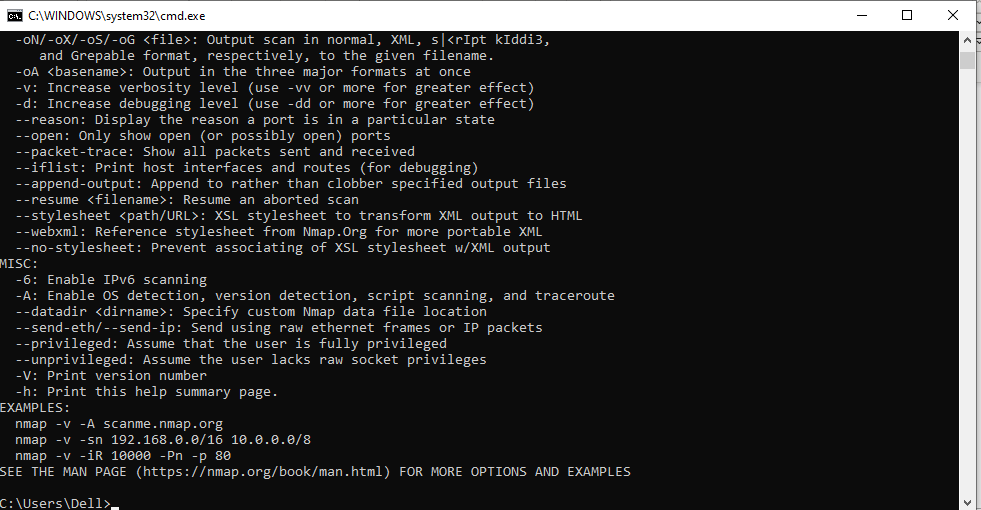
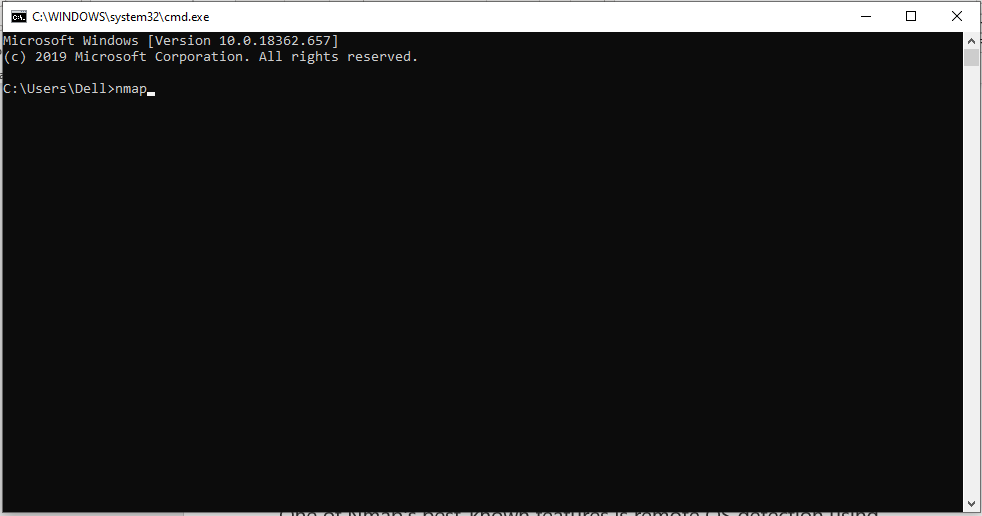
**Running nmap scan**

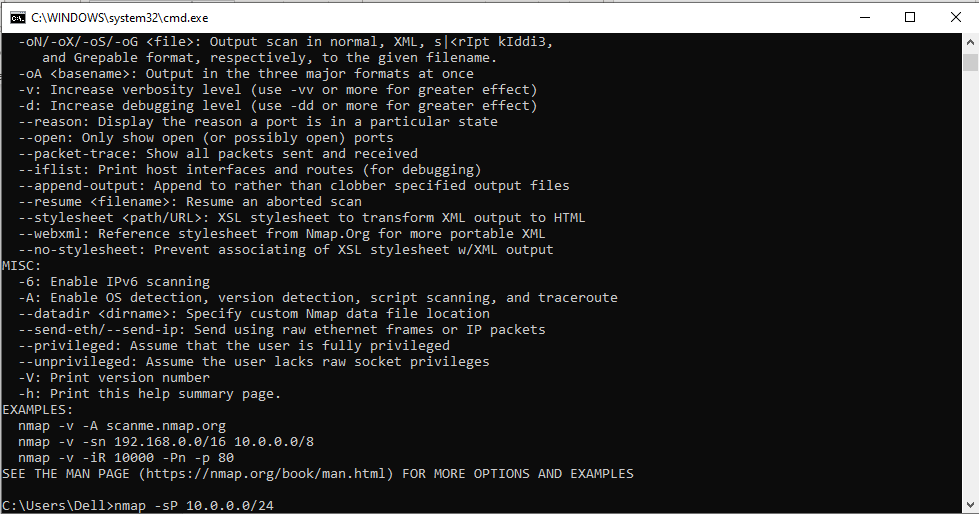
Nmap will scan a target System and report which ports are open and which are closed.

1)Download Nmap scan in Windows

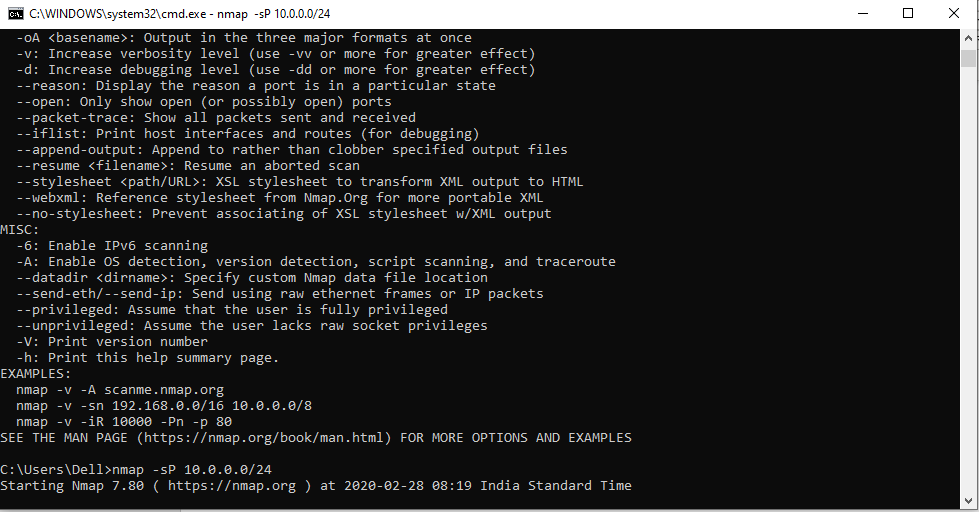
2) Copy and paste it in C:\Program Files\nmap-7.80

3)To run nmap scan,type nmap in command line window.(eg: nmap 172.16.1.34)



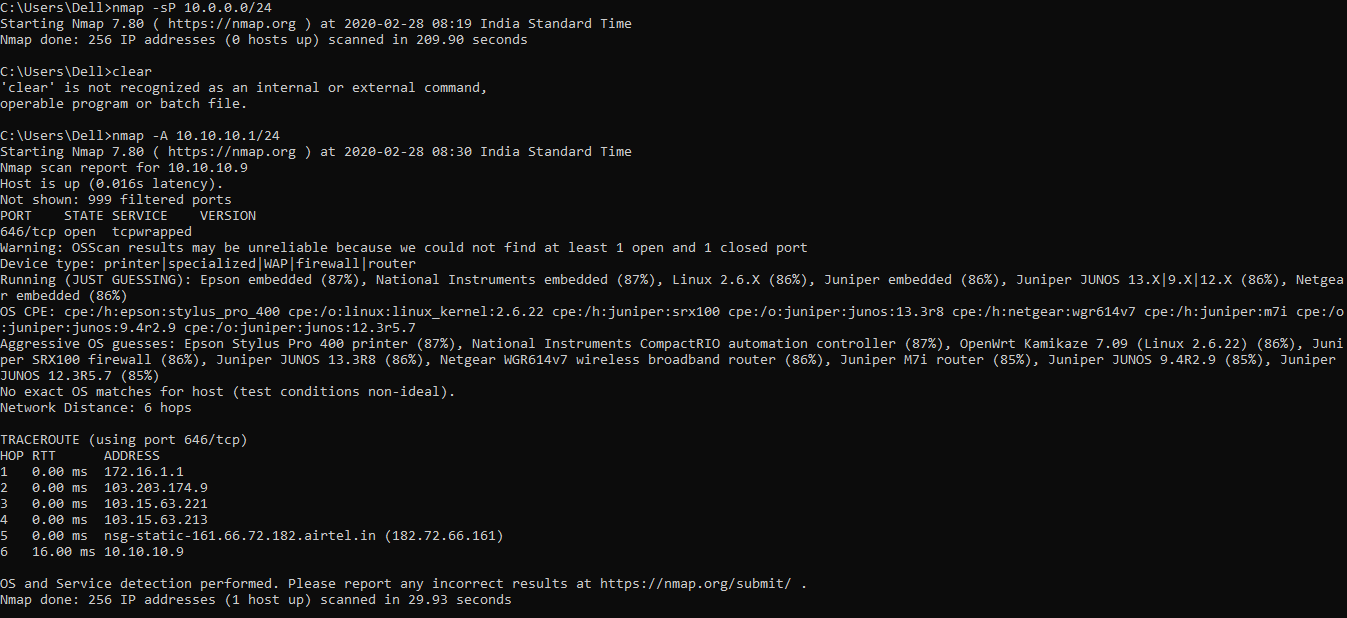


**The above screen shot command is used for discovering all live hosts on our network.**



**OS detection using Nmap**

One of Nmap's best-known features is remote OS detection using TCP/IP stack fingerprinting.



**Do the following using NS2 Simulator**

**i) NS2 Simulator Introduction**

Network Simulator (Version 2), widely known as NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks.

**a) Features of NS2**

1. It is a discrete event simulator for networking research.

2. It provides substantial support to simulate bunch of protocols like TCP, FTP,DSR etc.

3. It simulates wired and wireless network.

4. It is primarily Unix based.

5. Uses TCL as its scripting language.

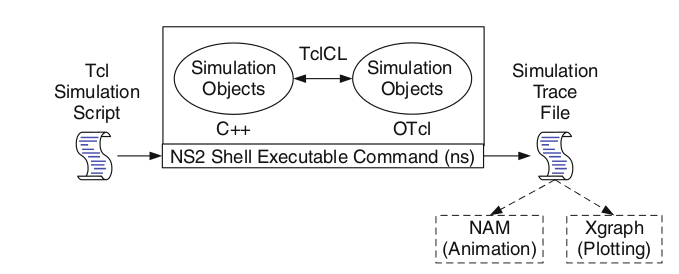
6. Otcl: Object oriented support

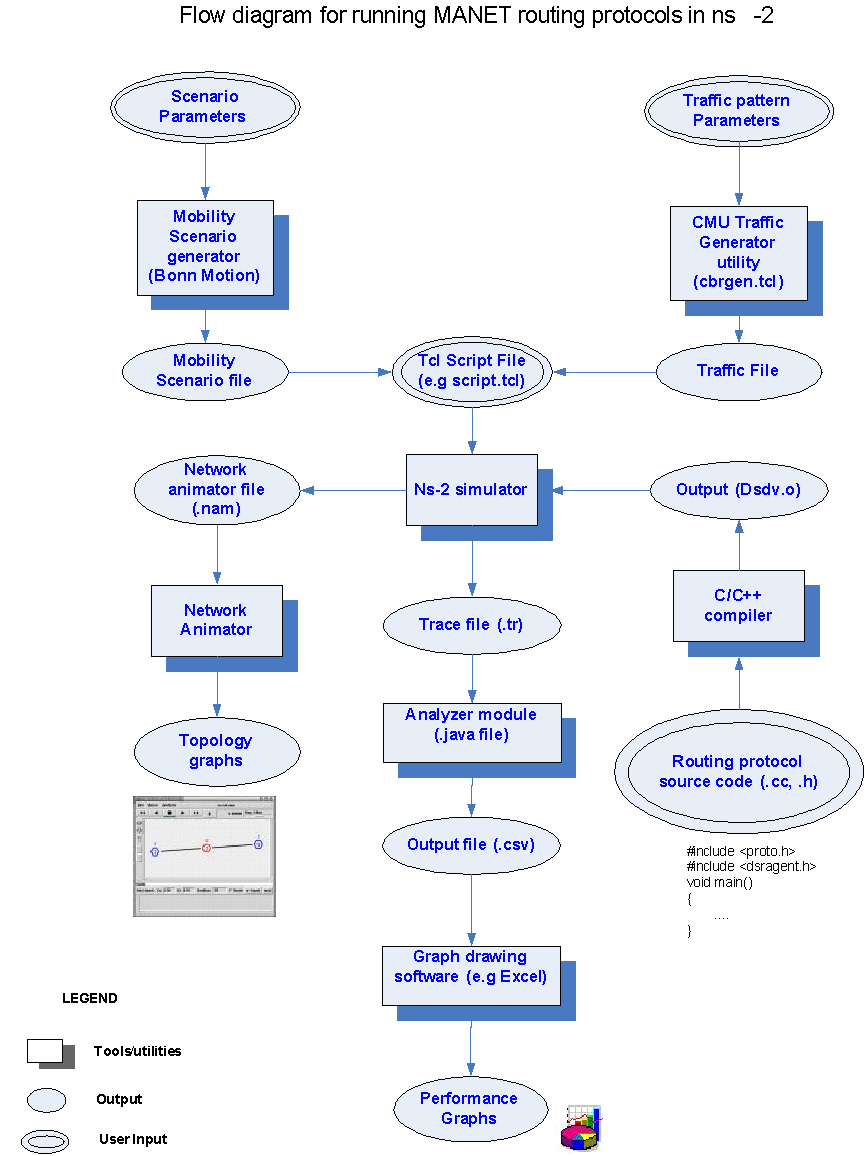
7. Tclcl: C++ and otcl linkage

8. Discrete event scheduler

**b) Basic Architecture**

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL.





**c) Trace file format is given below:**



In this format 12 fields and we can explain it as;

1. EVENT OR TYPE IDENTIFIER

+ :a packet enque event

- :a packet deque event

r :a packet reception event

d :a packet drop (e.g., sent to dropHead\_) event

c :a packet collision at the MAC level

2. TIME : at which the packet tracing string is created.  
3-4. SOURCE AND DESTINATION NODE : source and destination ID's of tracing objects.  
5. PACKET NAME : Name of the packet type.  
6. PACKET SIZE : Size of packet in bytes.  
7. FLAGS : 7 digit flag string.  
“-”: disable

1st = “E”: ECN (Explicit Congestion Notification) echo is enabled.

2nd = “P”: the priority in the IP header is enabled.

3rd : Not in use

4th = “A”: Congestion action

5th = “E”: Congestion has occurred.

6th = “F”: The TCP fast start is used.

7th = “N”: Explicit Congestion Notification (ECN) is on.

8. FLOW ID  
9-10. SOURCE AND DESTINATION ADDRESS : The format of these two fields is “a.b”, where “a" is the address and "b" is the port.  
11. SEQUENCE NUMBER  
12. PACKET UNIQUE ID

Snapshot of trace file(out.tr)

+ 1.04208 2 3 cbr 1000 ------- 0 1.0 3.1 4 6

- 1.04208 2 3 cbr 1000 ------- 0 1.0 3.1 4 6

r 1.04416 2 3 cbr 1000 ------- 0 1.0 3.1 3 5

+ 1.048 1 2 cbr 1000 ------- 0 1.0 3.1 6 10

- 1.048 1 2 cbr 1000 ------- 0 1.0 3.1 6 10

r 1.05008 1 2 cbr 1000 ------- 0 1.0 3.1 5 7

+ 1.05008 2 3 cbr 1000 ------- 0 1.0 3.1 5 7

- 1.05008 2 3 cbr 1000 ------- 0 1.0 3.1 5 7

r 1.050136 0 2 tcp 1540 ------- 0 0.0 3.0 1 8

+ 1.050136 2 3 tcp 1540 ------- 0 0.0 3.0 1 8

**ii ) NS2 Simulation for packet tranmission in wired networks**

**1)create tcpudp.tcl**

#===================================

# Simulation parameters setup

#===================================

set val(stop) 10.0; # time of simulation end

#===================================

# Initialization

#===================================

#Create a ns simulator

set ns [new Simulator]

#Open the NS trace file

set tracefile [open tcpudp.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open tcpudp.nam w]

$ns namtrace-all $namfile

#===================================

# Nodes Definition

#===================================

#Create 5 nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

#===================================

# Links Definition

#===================================

#Createlinks between nodes

$ns duplex-link $n2 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n2 $n1 50

$ns duplex-link $n1 $n0 100.0Mb 10ms DropTail

$ns queue-limit $n1 $n0 50

$ns duplex-link $n4 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n4 $n1 50

$ns duplex-link $n3 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n3 $n1 50

#Give node position (for NAM)

$ns duplex-link-op $n2 $n1 orient right-down

$ns duplex-link-op $n1 $n0 orient right

$ns duplex-link-op $n4 $n1 orient right-up

$ns duplex-link-op $n3 $n1 orient right

#===================================

# Agents Definition

#===================================

#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n3 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n0 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500

#Setup a UDP connection

set udp2 [new Agent/UDP]

$ns attach-agent $n4 $udp2

set null3 [new Agent/Null]

$ns attach-agent $n0 $null3

$ns connect $udp2 $null3

$udp2 set packetSize\_ 1500

#===================================

# Applications Definition

#===================================

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp2

$cbr1 set packetSize\_ 1000

$cbr1 set rate\_ 1.0Mb

$cbr1 set random\_ null

$ns at 1.0 "$cbr1 start"

$ns at 2.0 "$cbr1 stop"

#===================================

# Termination

#===================================

#Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile

exec nam tcpudp.nam &

exit 0

}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

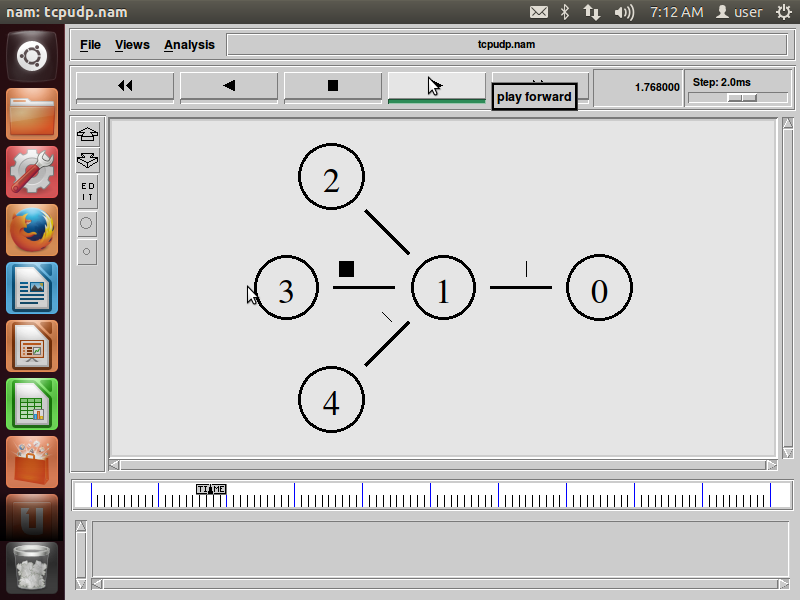
$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run

…………………………………………………………………………………………………………..

**2)Execute tcpudp.tcl using the following command**

$ns tcpudp.tcl

**3)It produces tcpudp.nam as output:**

**iii)Simulate to find the number of packets dropped in Packet transmission.**

set ns [new Simulator] # Letter S is capital

set nf [open PA1.nam w] # open a nam trace file in write mode

$ns namtrace-all $nf # nf nam filename

set tf [open PA1.tr w] # tf trace filename

$ns trace-all $tf

proc finish { } {

global ns nf tf $ns flush-trace # clears trace file contents

close $nf close $tf exec nam PA1.nam & exit 0

}

set n0 [$ns node] # creates 3 nodes

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 200Mb 10ms DropTail # establishing links

$ns duplex-link $n2 $n3 1Mb 1000ms DropTail

$ns queue-limit $n0 $n2 10

set udp0 [new Agent/UDP] # attaching transport layer protocols

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR] # attaching application layer protocols

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set null0 [new Agent/Null] # creating sink(destination) node

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$ns at 0.1 "$cbr0 start"

$ns at 1.0 "finish"

$ns run

**AWK file:** *(Open a new editor using “vi command” and write awk file and save with “.awk” extension)*

#immediately after BEGIN should open braces „{„

BEGIN { c=0;}

{

if($1= ="d")

{ c++;

printf("%s\t%s\n",$5,$11);

}

}

END { printf("The number of packets dropped =%d\n",c); }

**Steps for execution**

*Open vi editor and type program. Program name should have the extension “ .tcl ”*

*[root@localhost ~]# vi lab1.tcl*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Open vi editor and type awk program. Program name should have the extension “.awk ”*

*[root@localhost ~]# vi lab1.awk*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Run the simulation program*

*[root@localhost~]# ns lab1.tcl*

*Here “ns” indicates network simulator. We get the topology shown in the snapshot.*

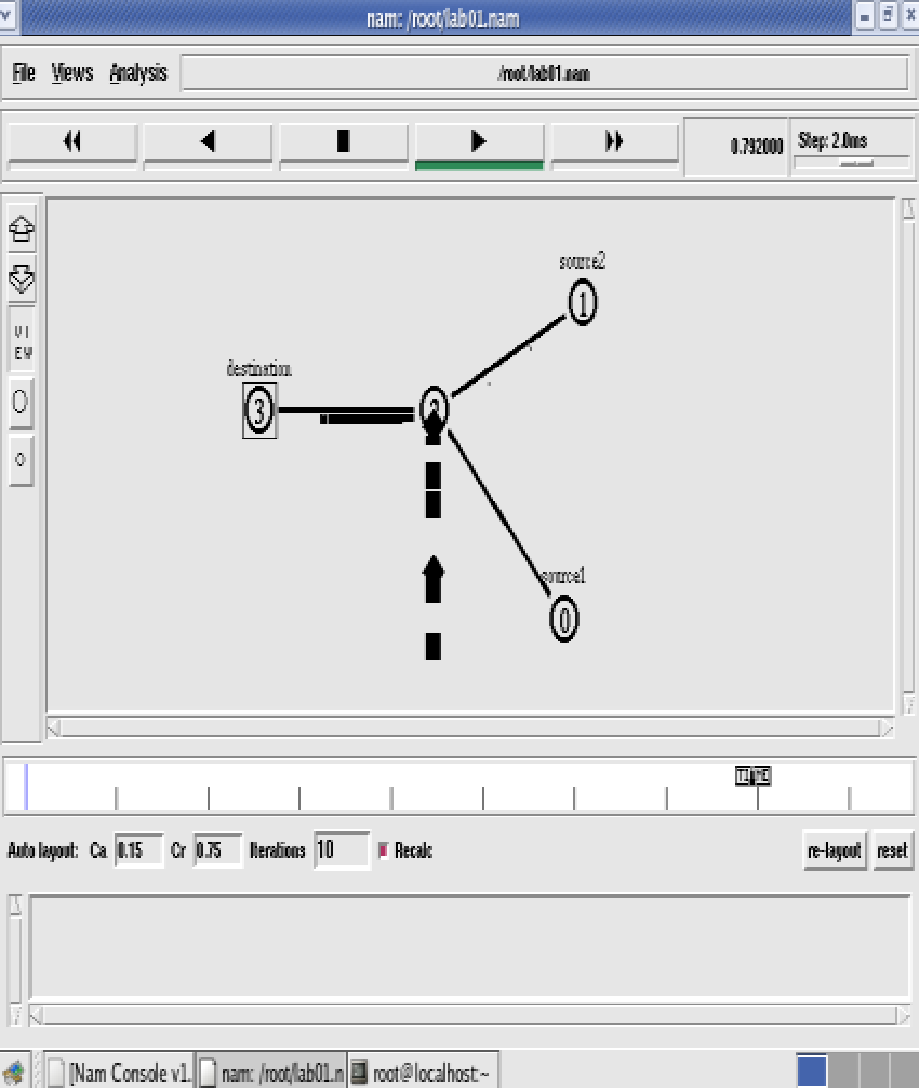
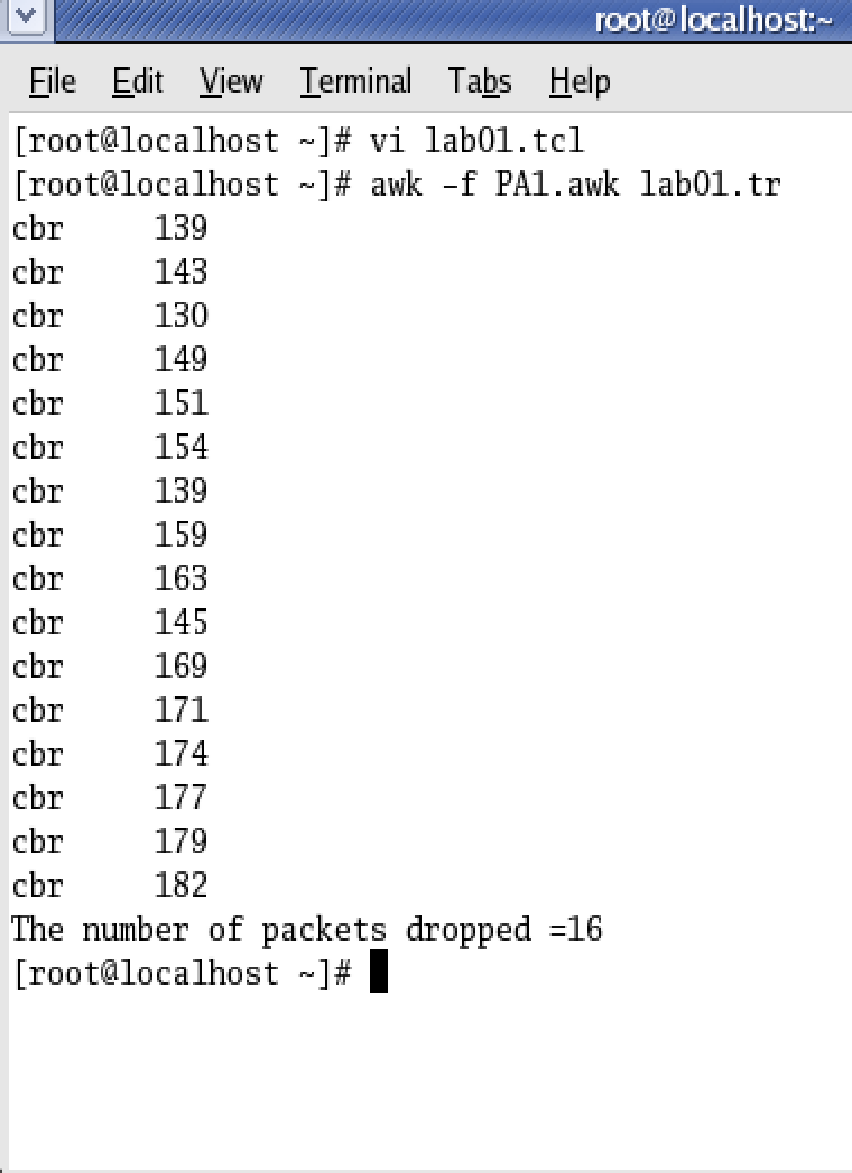
*Now press the play button in the simulation window and the simulation will begins.*

*After simulation is completed run awk file to see the output ,*

*[root@localhost~]# awk –f lab1.awk lab1.tr*

*To see the trace file contents open the file as ,*

*[root@localhost~]# vi lab1.tr*

****

***Output Topology***

**iv) Simulate to find number of packets dropped by tcp/udp**

set ns [new Simulator]

set nf [open lab2.nam w]

$ns namtrace-all $nf

set tf [open lab2.tr w]

$ns trace-all $tf

proc finish { }

{

global ns nf tf

$ns flush-trace

close $nf

close $tf

exec nam lab2.nam & exit 0

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

$ns duplex-link $n0 $n2 10Mb 1ms DropTail

$ns duplex-link $n1 $n2 10Mb 1ms DropTail

$ns duplex-link $n2 $n3 10Mb 1ms DropTail

set tcp0 [new Agent/TCP] # letters A,T,C,P are capital

$ns attach-agent $n0 $tcp0

set udp1 [new Agent/UDP] # letters A,U,D,P are capital

$ns attach-agent $n1 $udp1

set null0 [new Agent/Null] # letters A and N are capital

$ns attach-agent $n3 $null0

set sink0 [new Agent/TCPSink] # letters A,T,C,P,S are capital

$ns attach-agent $n3 $sink0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp1

$ns connect $tcp0 $sink0

$ns connect $udp1 $null0

$ns at 0.1 "$cbr1 start"

$ns at 0.2 "$ftp0 start"

$ns at 0.5 "finish"

$ns run

**AWK file:** *(Open a new editor using “vi command” and write awk file and save with “.awk” extension)*

BEGIN{

udp=0;

tcp=0;

}

{

if($1= = “r” && $5 = = “cbr”)

{

udp++;

}

else if($1 = = “r” && $5 = = “tcp”)

{

tcp++;

}

}

END{

printf(“Number of packets sent by TCP = %d\n”, tcp);

printf(“Number of packets sent by UDP=%d\n”,udp);

}

**Steps for execution:**

*Open vi editor and type program. Program name should have the extension “ .tcl ”*

*[root@localhost ~]# vi lab2.tcl*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Open vi editor and type awk program. Program name should have the extension “.awk ”*

*[root@localhost ~]# vi lab2.awk*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Run the simulation program*

*[root@localhost~]# ns lab2.tcl*

o *Here “ns” indicates network simulator. We get the topology shown in the snapshot.*

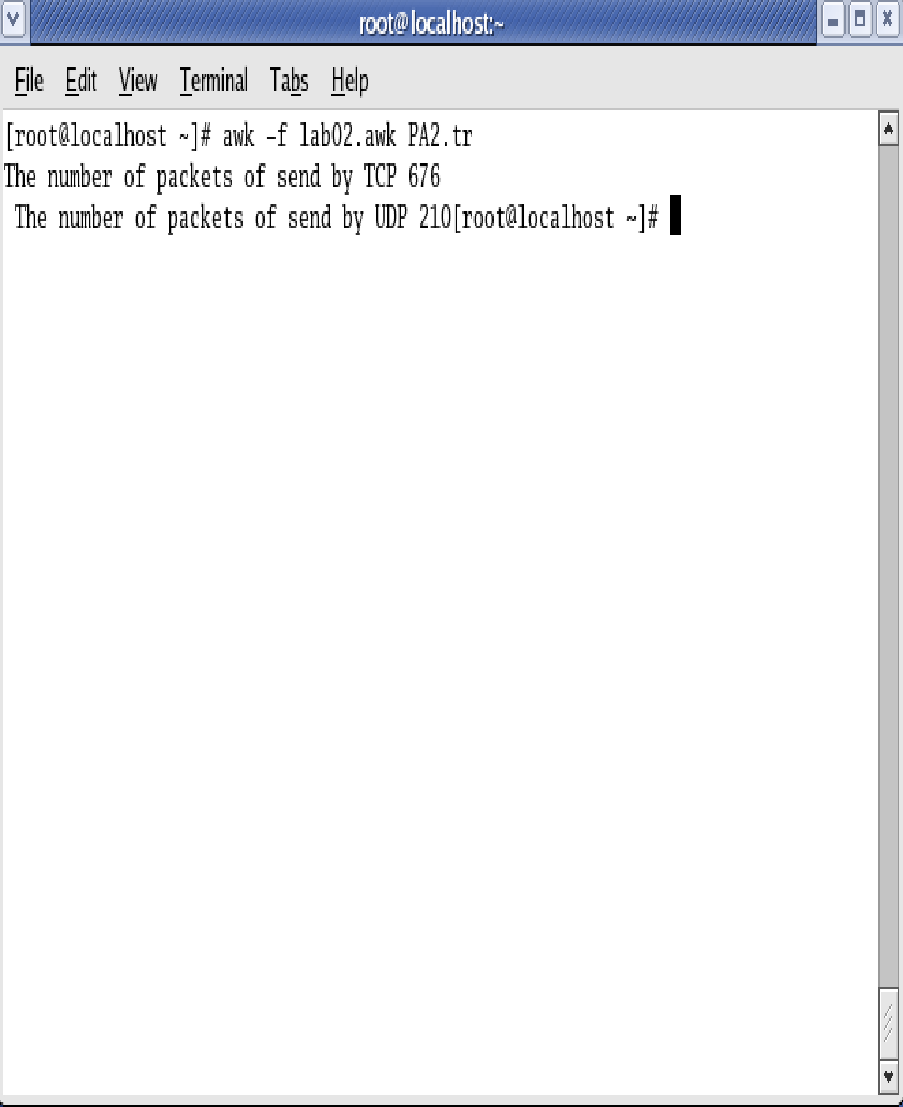
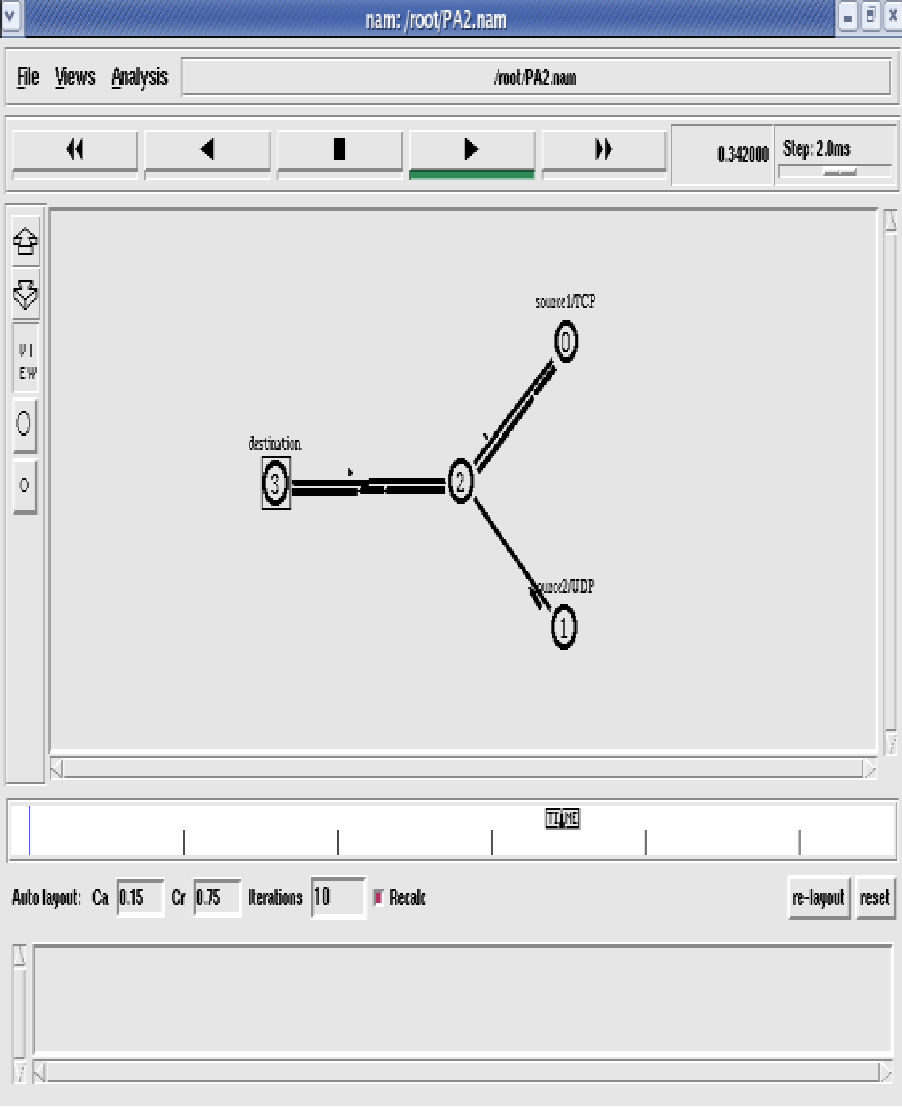
o *Now press the play button in the simulation window and the simulation will begins.*

*After simulation is completed run awk file to see the output ,*

*[root@localhost~]# awk –f lab2.awk lab2.tr*

*To see the trace file contents open the file as ,*

*[root@localhost~]# vi lab2.tr*

****

**Topology Output**

v) Simulate to find number of packets dropped due to congestion

set nf [ open lab4.nam w ]

$ns namtrace-all $nf

set tf [ open lab4.tr w ]

$ns trace-all $tf

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns duplex-link $n0 $n4 1005Mb 1ms DropTail

$ns duplex-link $n1 $n4 50Mb 1ms DropTail

$ns duplex-link $n2 $n4 2000Mb 1ms DropTail

$ns duplex-link $n3 $n4 200Mb 1ms DropTail

$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set p1 [new Agent/Ping] # letters A and P should be capital

$ns attach-agent $n0 $p1

$p1 set packetSize\_ 50000

$p1 set interval\_ 0.0001

set p2 [new Agent/Ping] # letters A and P should be capital

$ns attach-agent $n1 $p2

set p3 [new Agent/Ping] # letters A and P should be capital

$ns attach-agent $n2 $p3

$p3 set packetSize\_ 30000

$p3 set interval\_ 0.00001

set p4 [new Agent/Ping] # letters A and P should be capital

$ns attach-agent $n3 $p4

set p5 [new Agent/Ping] # letters A and P should be capital

$ns attach-agent $n5 $p5

$ns queue-limit $n0 $n4 5

$ns queue-limit $n2 $n4 3

$ns queue-limit $n4 $n5 2

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id]received answer from $from with round trip time $rtt msec"

}

# please provide space between $node\_ and id. No space between $ and from. No space between and $ and rtt \*/

$ns connect $p1 $p5

$ns connect $p3 $p4

proc finish { } {

global ns nf tf

$ns flush-trace

close $nf

close $tf

exec nam lab4.nam &

exit 0

}

$ns at 0.1 "$p1 send"

$ns at 0.2 "$p1 send"

$ns at 0.3 "$p1 send"

$ns at 0.4 "$p1 send"

$ns at 0.5 "$p1 send"

$ns at 0.6 "$p1 send"

$ns at 0.7 "$p1 send"

$ns at 0.8 "$p1 send"

$ns at 0.9 "$p1 send"

$ns at 1.0 "$p1 send"

$ns at 1.1 "$p1 send"

$ns at 1.2 "$p1 send"

$ns at 1.3 "$p1 send"

$ns at 1.4 "$p1 send"

$ns at 1.5 "$p1 send"

$ns at 1.6 "$p1 send"

$ns at 1.7 "$p1 send"

$ns at 1.8 "$p1 send"

$ns at 1.9 "$p1 send"

$ns at 2.0 "$p1 send"

$ns at 2.1 "$p1 send"

$ns at 2.2 "$p1 send"

$ns at 2.3 "$p1 send"

$ns at 2.4 "$p1 send"

$ns at 2.5 "$p1 send"

$ns at 2.6 "$p1 send"

$ns at 2.7 "$p1 send"

$ns at 2.8 "$p1 send"

$ns at 2.9 "$p1 send"

$ns at 0.1 "$p3 send"

$ns at 0.2 "$p3 send"

$ns at 0.3 "$p3 send"

$ns at 0.4 "$p3 send"

$ns at 0.5 "$p3 send"

$ns at 0.6 "$p3 send"

$ns at 0.7 "$p3 send"

$ns at 0.8 "$p3 send"

$ns at 0.9 "$p3 send"

$ns at 1.0 "$p3 send"

$ns at 1.1 "$p3 send"

$ns at 1.2 "$p3 send"

$ns at 1.3 "$p3 send"

$ns at 1.4 "$p3 send"

$ns at 1.5 "$p3 send"

$ns at 1.6 "$p3 send"

$ns at 1.7 "$p3 send"

$ns at 1.8 "$p3 send"

$ns at 1.9 "$p3 send"

$ns at 2.0 "$p3 send"

$ns at 2.1 "$p3 send"

$ns at 2.2 "$p3 send"

$ns at 2.3 "$p3 send"

$ns at 2.4 "$p3 send"

$ns at 2.5 "$p3 send"

$ns at 2.6 "$p3 send"

$ns at 2.7 "$p3 send"

$ns at 2.8 "$p3 send"

$ns at 2.9 "$p3 send"

$ns at 3.0 "finish"

$ns run

**AWK file:** *(Open a new editor using “vi command” and write awk file and save with “.awk” extension)*

BEGIN{

drop=0;

}

{

if($1= ="d" )

{

drop++;

}

}

END{

printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);

}

**Steps for execution:**

*1) Open vi editor and type program. Program name should have the extension “ .tcl ”*

*[root@localhost ~]# vi lab4.tcl*

*2) Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*3) Open vi editor and type awk program. Program name should have the extension “.awk ”*

*[root@localhost ~]# vi lab4.awk*

*4) Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*5) Run the simulation program*

*[root@localhost~]# ns lab4.tcl*

*i) Here “ns” indicates network simulator. We get the topology shown in the snapshot.*

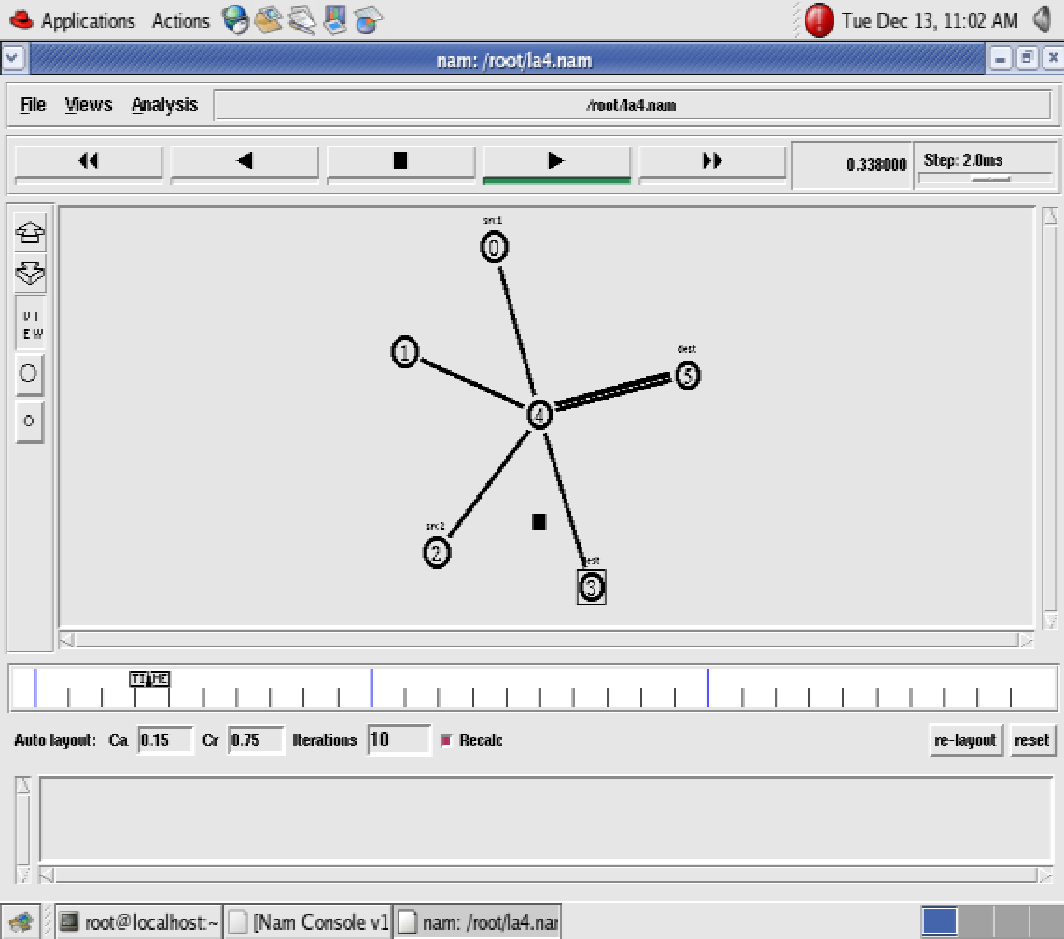
*ii) Now press the play button in the simulation window and the simulation will begins.*

*6) After simulation is completed run awk file to see the output ,*

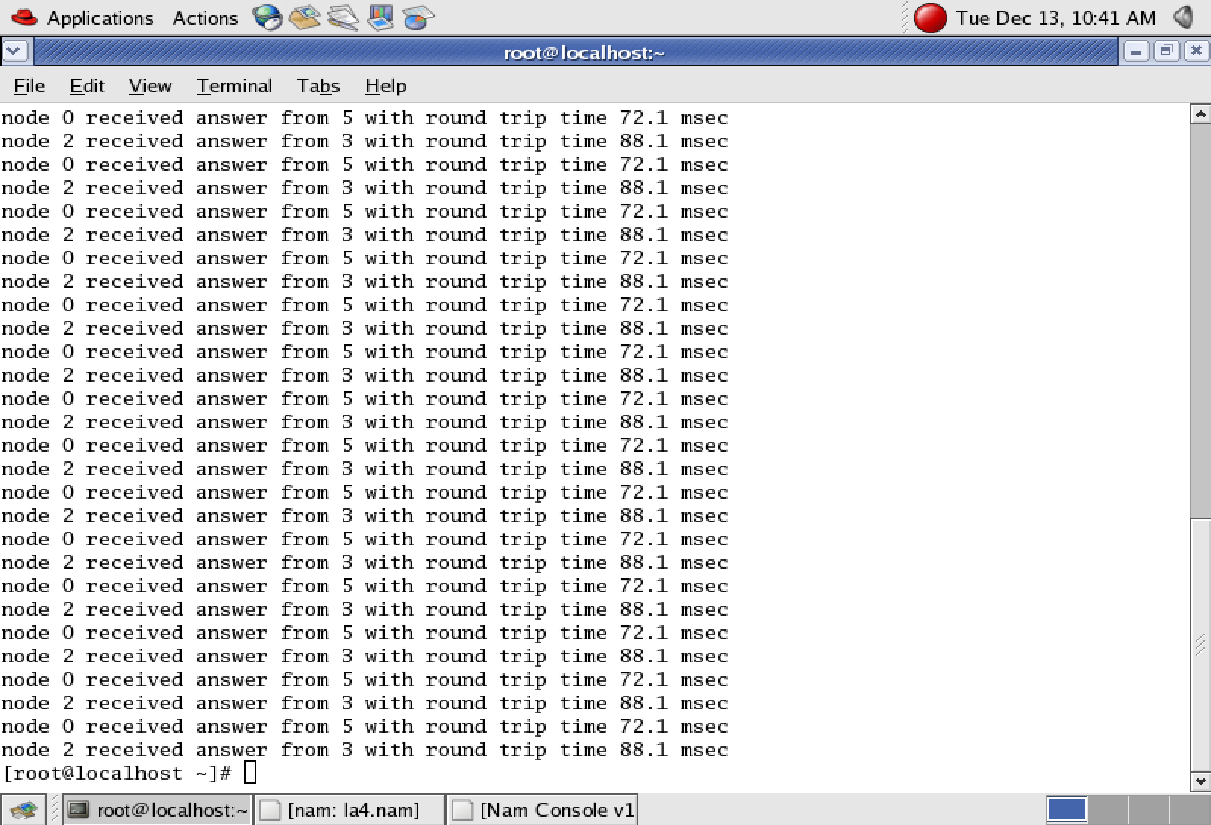
*[root@localhost~]# awk –f lab4.awk lab4.tr*

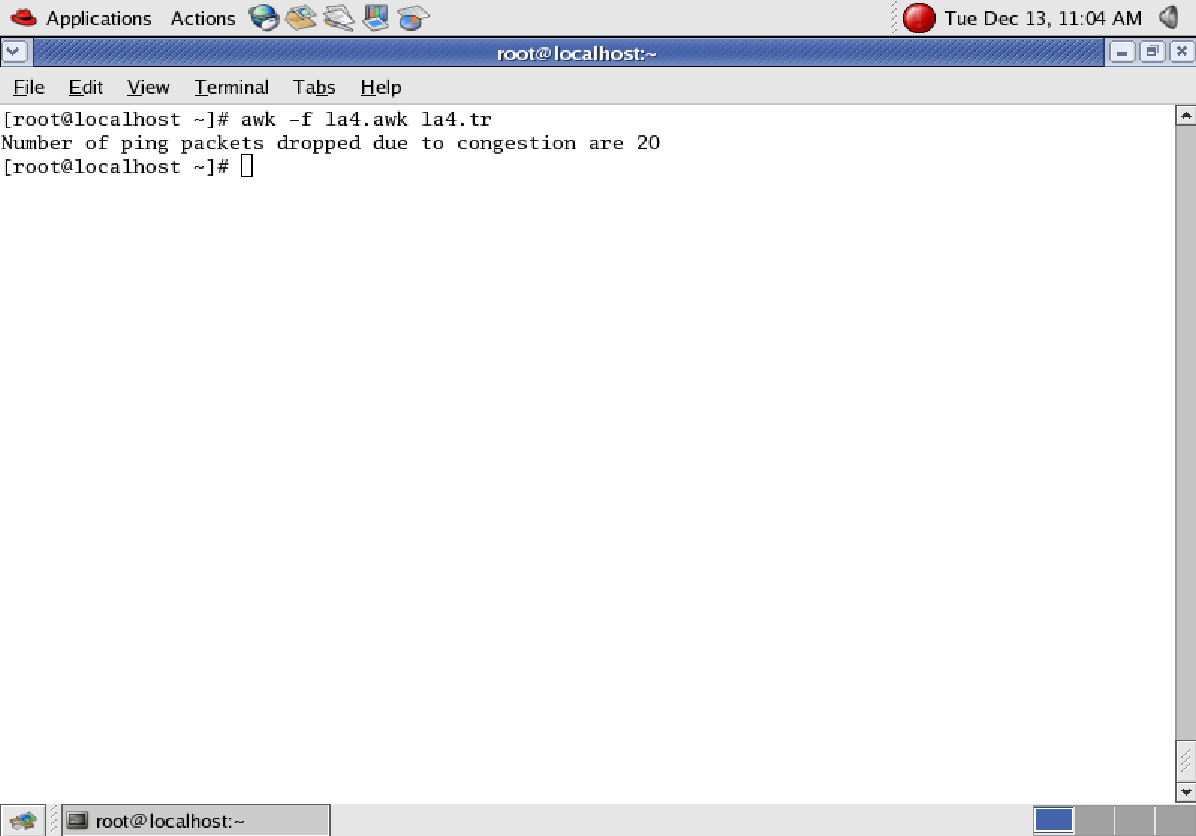
*7) To see the trace file contents open the file as ,*

*[root@localhost~]# vi lab4.tr*

****

**Topology**

****

****

**Output Screens**

**vi) Simulate to plot Congestion window for different Source/Destination**

set ns [new Simulator]

set tf [open pgm7.tr w]

$ns trace-all $tf

set nf [open pgm7.nam w]

$ns namtrace-all $nf

set n0 [$ns node] $n0 color "magenta" $n0 label "src1"

set n1 [$ns node]

set n2 [$ns node] $n2 color "magenta" $n2 label "src2"

set n3 [$ns node] $n3 color "blue" $n3 label "dest2"

set n4 [$ns node]

set n5 [$ns node] $n5 color "blue" $n5 label "dest1"

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/ DropTail Mac/802\_3 # should come in single line

$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ftp0 set packetSize\_ 500

$ftp0 set interval\_ 0.0001

set sink5 [new Agent/TCPSink]

$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]

$ns attach-agent $n2 $tcp2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ftp2 set packetSize\_ 600

$ftp2 set interval\_ 0.001

set sink3 [new Agent/TCPSink]

$ns attach-agent $n3 $sink3

$ns connect $tcp2 $sink3

set file1 [open file1.tr w]

$tcp0 attach $file1

set file2 [open file2.tr w]

$tcp2 attach $file2

$tcp0 trace cwnd\_ # must put underscore ( \_ ) after cwnd and no space between them

$tcp2 trace cwnd\_

proc finish { }

{

global ns nf tf

$ns flush-trace close

$tf close

$nf exec nam pgm7.nam & exit 0

}

$ns at 0.1 "$ftp0 start"

$ns at 5 "$ftp0 stop"

$ns at 7 "$ftp0 start"

$ns at 0.2 "$ftp2 start"

$ns at 8 "$ftp2 stop"

$ns at 14 "$ftp0 stop"

$ns at 10 "$ftp2 start"

$ns at 15 "$ftp2 stop"

$ns at 16 "finish"

$ns run

**AWK file:** *(Open a new editor using “vi command” and write awk file and save with “.awk” extension)*

cwnd:- means congestion window

BEGIN {

}

{

if($6= ="cwnd\_") # don‟t leave space after writing cwnd\_

printf("%f\t%f\t\n",$1,$7); # you must put \n in printf

}

END {

}

Steps for execution

*Open vi editor and type program. Program name should have the extension “ .tcl ”*

*[root@localhost ~]# vi lab7.tcl*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Open vi editor and type awk program. Program name should have the extension “.awk ”*

*[root@localhost ~]# vi lab7.awk*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Run the simulation program*

*[root@localhost~]# ns lab7.tcl*

*After simulation is completed run awk file to see the output ,*

*[root@localhost~]# awk –f lab7.awk file1.tr > a1*

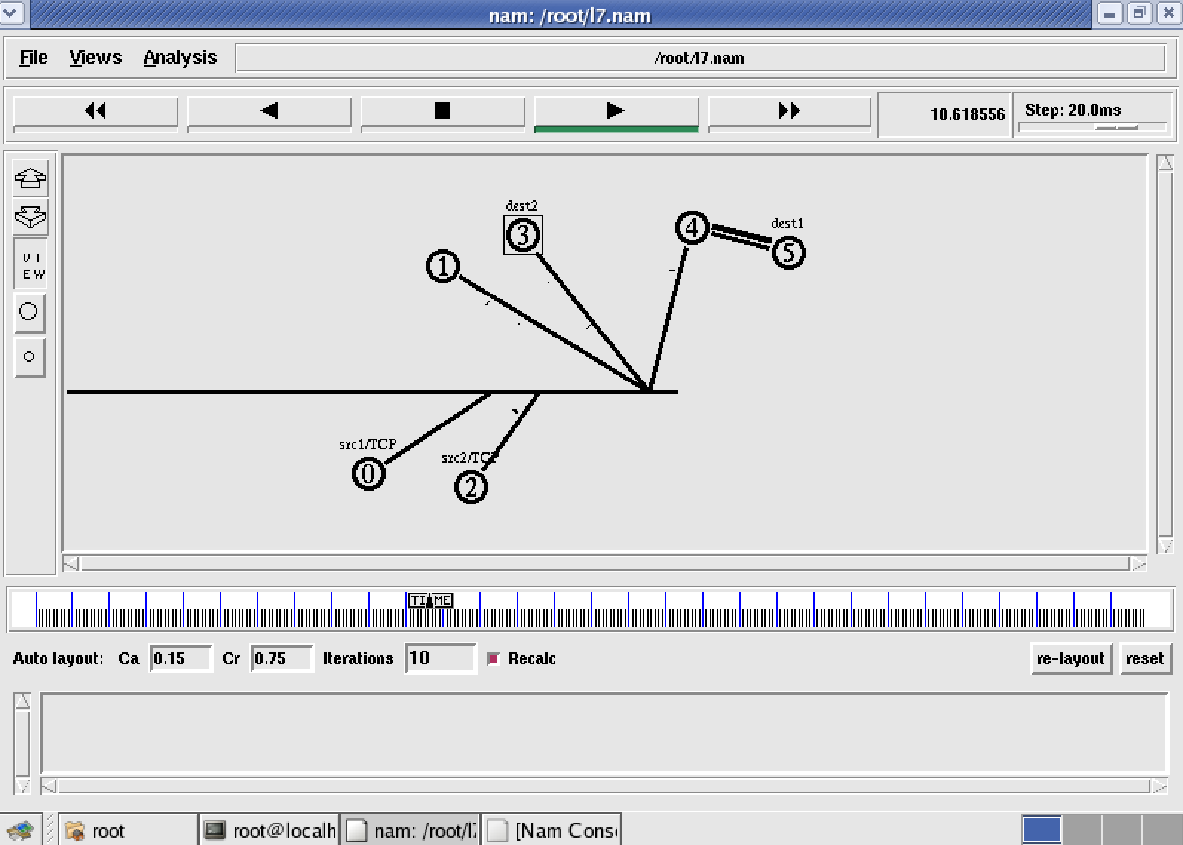
*[root@localhost~]# awk –f lab7.awk file2.tr > a2*

*[root@localhost~]# xgraph a1 a2*

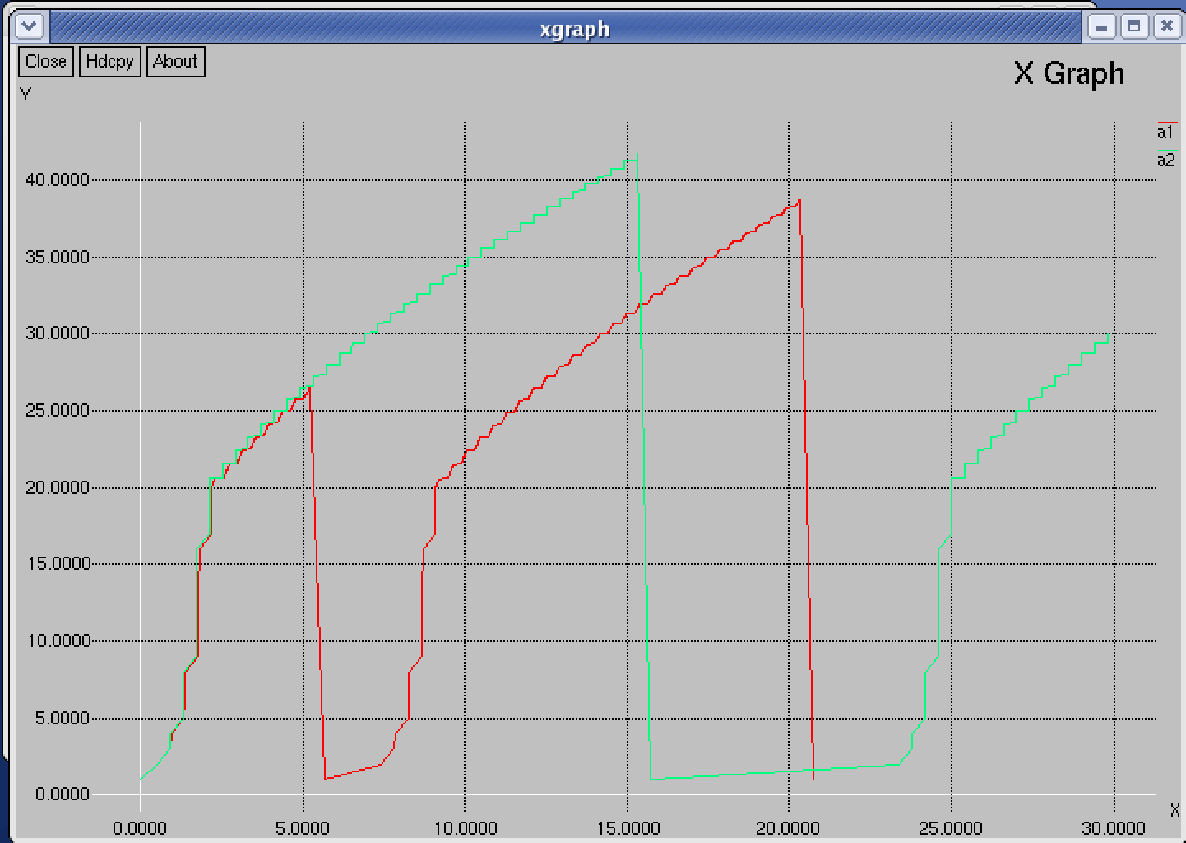
*Here we are using the congestion window trace files i.e. file1.tr and file2.tr and we are redirecting the contents of those files to new files say a1 and a2 using output redirection operator (>).*

*To see the trace file contents open the file as ,*

*[root@localhost~]# vi lab7.tr*

****

**Topology**

****

**XGraph Output**

**vii) Simulate to Determine the Performance with respect to Transmission of Packets in a wireless network (with 3 nodes)**

set ns [new Simulator]

set tf [open lab8.tr w]

$ns trace-all $tf

set topo [new Topography]

$topo load\_flatgrid 1000 1000

set nf [open lab8.nam w]

$ns namtrace-all-wireless $nf 1000 1000

$ns node-config -adhocRouting DSDV \

-llType LL \

-macType Mac/802\_11 \

-ifqType Queue/DropTail \

-ifqLen 50 \

-phyType Phy/WirelessPhy \

-channelType Channel/WirelessChannel \

-prrootype Propagation/TwoRayGround \

-antType Antenna/OmniAntenna \

-topoInstance $topo \

-agentTrace ON \

-routerTrace ON

create-god 3

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

$n0 label "tcp0"

$n1 label "sink1/tcp1"

$n2 label "sink2"

$n0 set X\_ 50

$n0 set Y\_ 50

$n0 set Z\_ 0

$n1 set X\_ 100

$n1 set Y\_ 100

$n1 set Z\_ 0

$n2 set X\_ 600

$n2 set Y\_ 600

$n2 set Z\_ 0

$ns at 0.1 "$n0 setdest 50 50 15"

$ns at 0.1 "$n1 setdest 100 100 25"

$ns at 0.1 "$n2 setdest 600 600 25"

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n1 $sink1

$ns connect $tcp0 $sink1

set tcp1 [new Agent/TCP]

$ns attach-agent $n1 $tcp1

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

set sink2 [new Agent/TCPSink]

$ns attach-agent $n2 $sink2

$ns connect $tcp1 $sink2

$ns at 5 "$ftp0 start"

$ns at 5 "$ftp1 start"

$ns at 100 "$n1 setdest 550 550 15"

$ns at 190 "$n1 setdest 70 70 15"

proc finish { } {

global ns nf tf

$ns flush-trace

exec nam lab8.nam &

close $tf

exit 0

}

$ ns at 250 "finish"

$ns run

**AWK file:** *(Open a new editor using “vi command” and write awk file and save with “.awk” extension)*

BEGIN{

count1=0

count2=0

pack1=0

pack2=0

time1=0

time2=0

}

{

if($1= ="r"&& $3= ="\_1\_" && $4= ="AGT")

{

count1++

pack1=pack1+$8

time1=$2

}

if($1= ="r" && $3= ="\_2\_" && $4= ="AGT")

{

count2++

pack2=pack2+$8

time2=$2

}

}

END{

printf("The Throughput from n0 to n1: %f Mbps \n‖, ((count1\*pack1\*8)/(time1\*1000000)));

printf("The Throughput from n1 to n2: %f Mbps", ((count2\*pack2\*8)/(time2\*1000000)));

}

**Steps for execution**

***O****pen vi editor and type program. Program name should have the extension “ .tcl ”*

*[root@localhost ~]# vi lab8.tcl*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Open vi editor and type awk program. Program name should have the extension “.awk ”*

*[root@localhost ~]# vi lab8.awk*

*Save the program by pressing “ESC key” first, followed by “Shift and :” keys simultaneously and type “wq” and press Enter key.*

*Run the simulation program*

*[root@localhost~]# ns lab8.tcl*

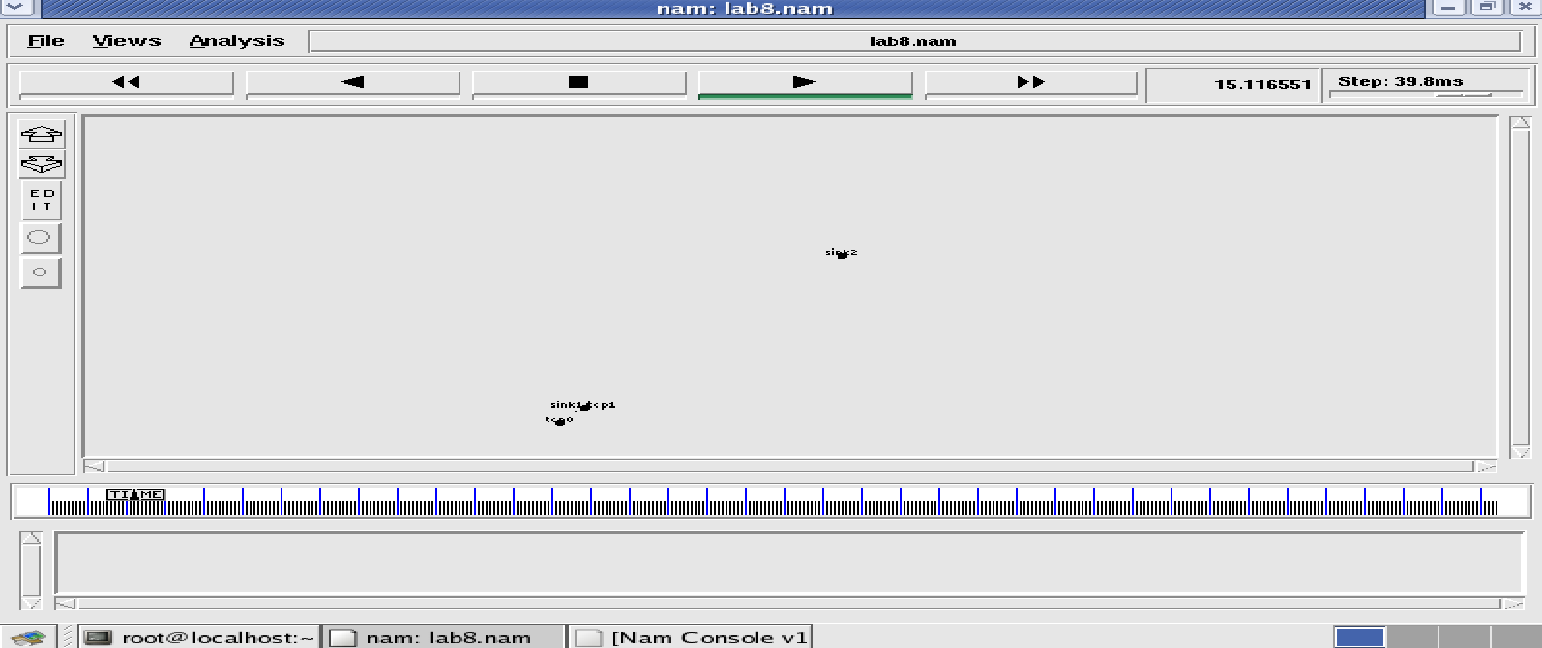
*Here “ns” indicates network simulator. We get the topology shown in the snapshot.*

o *Now press the play button in the simulation window and the simulation will begins.*

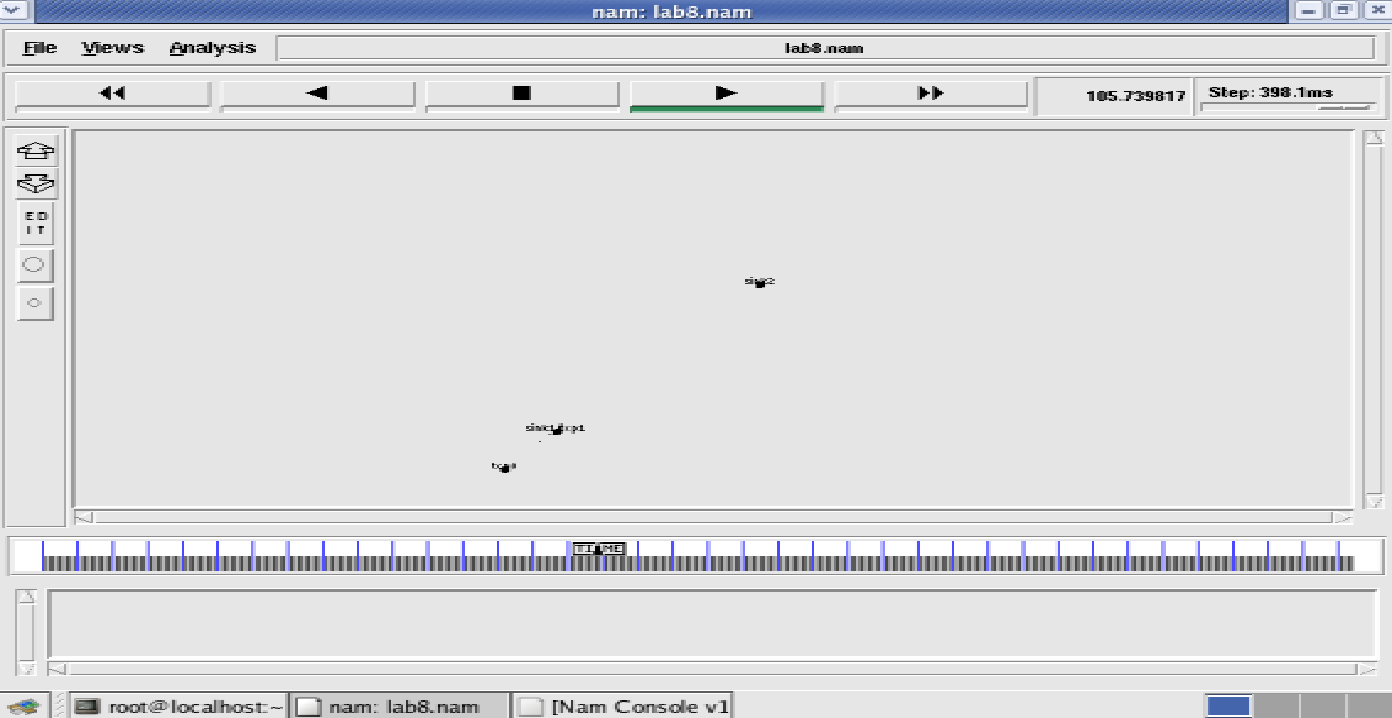
*After simulation is completed run awk file to see the output ,*

*[root@localhost~]# awk –f lab8.awk lab8.tr*

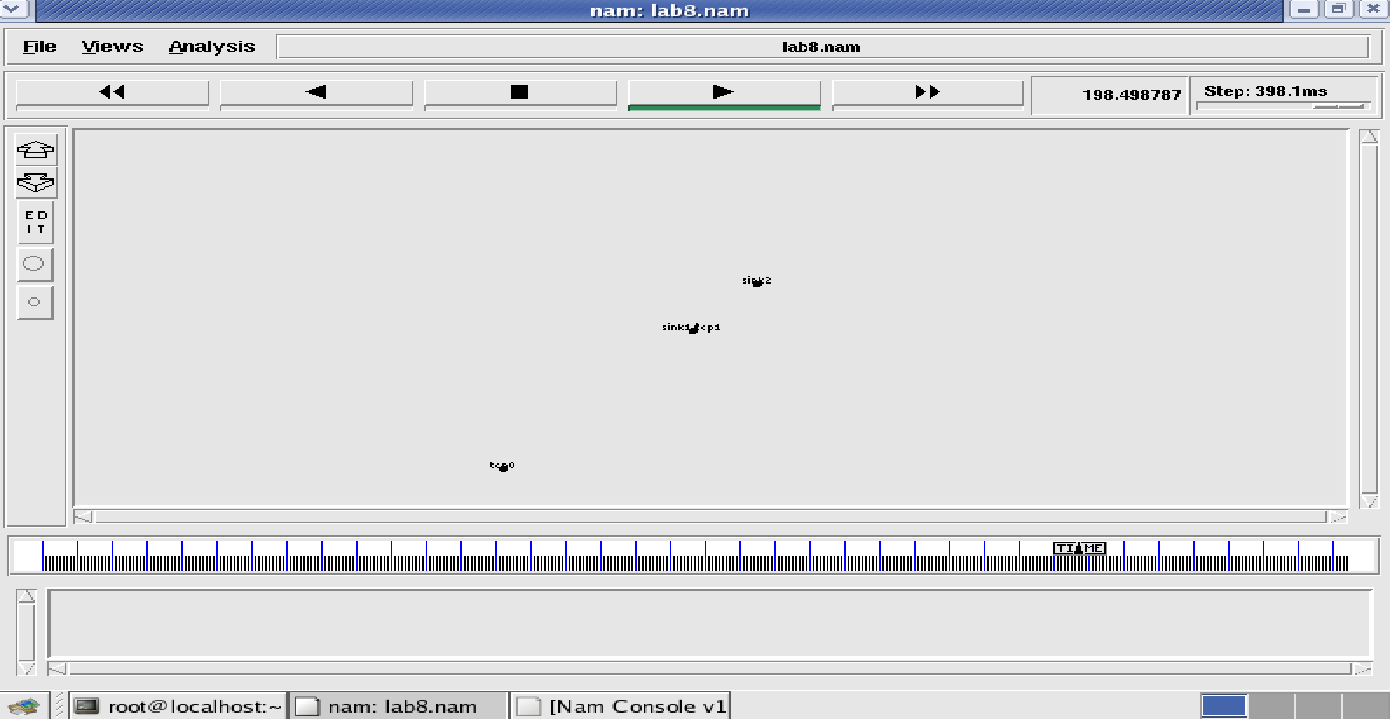
**Output:**

****

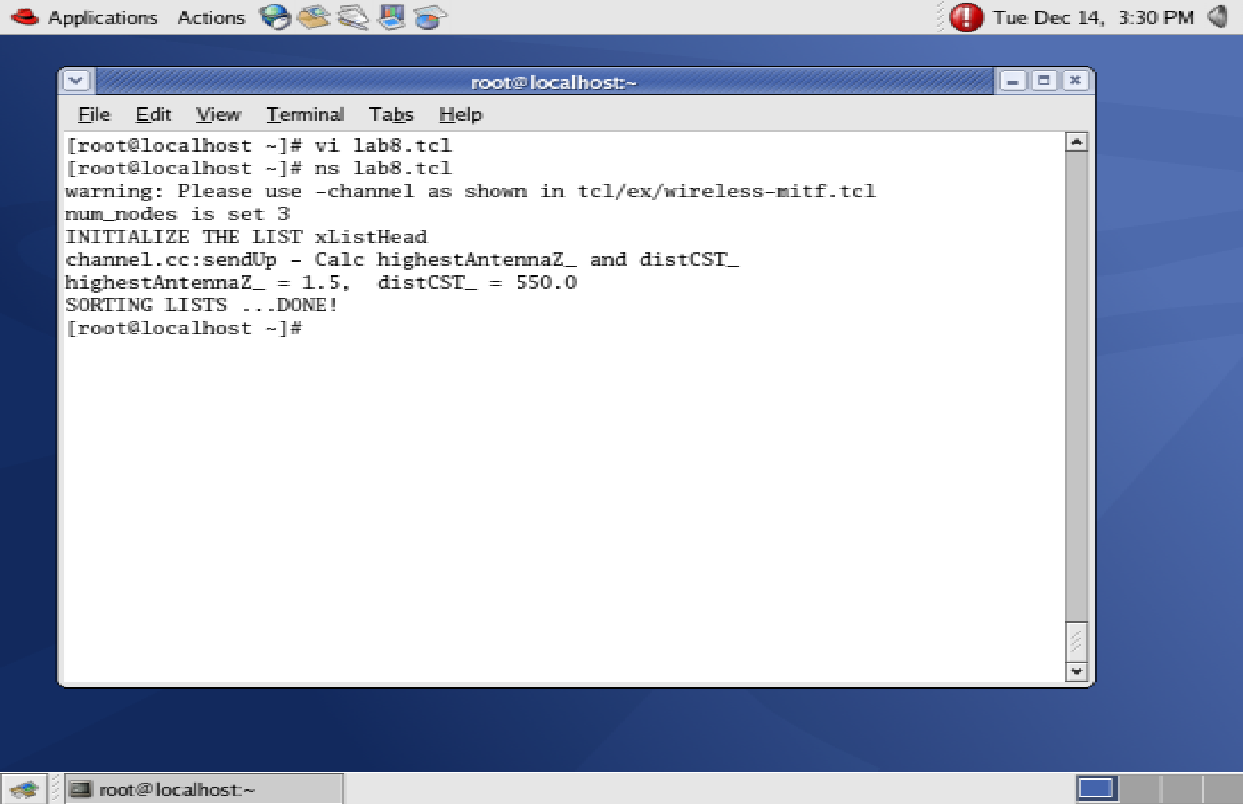
**Node 1 and 2 are communicating**

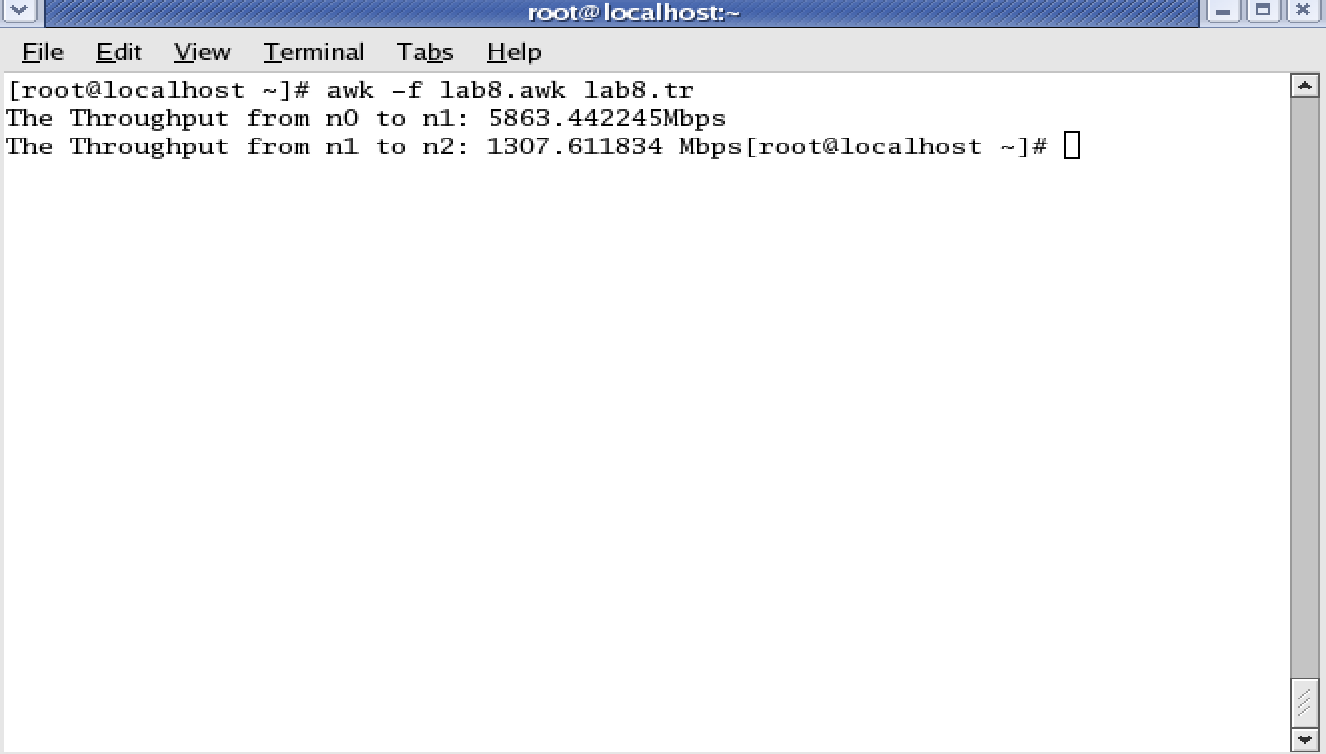
****

**Node 2 is moving towards node 3**

****

**Node 2 is coming back from node 3 towards node1**

****

****

**Additional Lab Programs**

CN-**1. Write a program for Hamming Code generation for error detection and correction**

Program

#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

#include<stdio.h>

char data[5];

int encoded[8], edata[7], syndrome[3];

int hmatrix[3][7]= { 1,0,0,0,1,1,1,

0,1,0,1,0,1,1,

0,0,1,1,1,0,1};

char gmatrix[4][8]={ "0111000", "1010100", "1100010", "1110001"};Computer Networks Lab Manual

DBIT

void main() {

int i,j;

clrscr();

cout<<"Hamming Code --- Encoding\n";

cout<<"Enter 4 bit data : ";

cin>>data;

cout<<"Generator Matrix\n";

for(i=0;i<4;i++) cout<<"\t"<<gmatrix[i]<<"\n";

cout<<"Encoded Data : ";

for(i=0;i<7;i++) {

for(j=0;j<4;j++)

encoded[i]+=((data[j]- '0')\*(gmatrix[j][i]- '0'));

encoded[i]=encoded[i]%2;

cout<<encoded[i]<<" ";

}

cout<<"\nHamming code --- Decoding\n";

cout<<"Enter Encoded bits as received : ";

for(i=0;i<7;i++) cin>>edata[i];

for(i=0;i<3;i++) {

for(j=0;j<7;j++)

syndrome[i]=syndrome[i]+(edata[j]\*hmatrix[i][j]);

syndrome[i]=syndrome[i]%2;

}

for(j=0;j<7;j++)

if ((syndrome[0]==hmatrix[0][j])&&(syndrome[1]==hmatrix[1][j])&&

(syndrome[2]==hmatrix[2][j]))

break;

if(j==7)

cout<<"Data is error free!!\n";

else {

cout<<"Error received at bit number "<<j+1<<" of the data\n";

edata[j]=!edata[j];

cout<<"The Correct data Should be : ";

for(i=0;i<7;i++) cout<<edata[i]<<" ";

}

}

}

Output

Hamming Code --- Encoding

Enter 4 bit data : 1 0 1 0

Generator Matrix

0111000

1010100

1100010

1110001

Encoded Data : 1 0 1 1 0 1 0

Hamming code --- Decoding

Enter Encoded bits as received : 1 0 1 1 0 1 1

Error received at bit number 7 of the data

The Correct data Should be : 1 0 1 1 0 1 0Computer Networks Lab Manual

Hamming Code --- Encoding

Enter 4 bit data : 1 0 1 0

Generator Matrix

0111000

1010100

1100010

1110001

Encoded Data : 1 0 1 1 0 1 0

Hamming code --- Decoding

Enter Encoded bits as received : 1 0 1 1 0 1 0

Data is error free!!

**CN-2.Implement the above program using message queues or FIFO as IPC channels.**

rProgram

/\*Server\*/

#include<stdio.h>

#include<unistd.h>

#include<sys/stat.h>

#include<fcntl.h>

#include<string.h>

#define FIFO1 "fifo1"

#define FIFO2 "fifo2"

#define PERMS 0666

char fname[256];

int main() {

int readfd, writefd, fd;

ssize\_t n;

char buff[512];

if (mkfifo(FIFO1, PERMS)<0)

printf("Cant Create FIFO Files\n");

if (mkfifo(FIFO2, PERMS)<0)

printf("Cant Create FIFO Files\n");

printf("Waiting for connection Request..\n");

readfd =open(FIFO1, O\_RDONLY, 0);

writefd=open(FIFO2, O\_WRONLY, 0);

printf("Connection Established..\n");

read(readfd, fname, 255);

printf("Client has requested file %s\n", fname);

if ((fd=open(fname,O\_RDWR))<0) {

strcpy(buff,"File does not exist..\n");

write(writefd, buff, strlen(buff));

} else {

while((n=read(fd, buff,512))>0)

write(writefd, buff, n);

}

close(readfd); unlink(FIFO1);

close(writefd); unlink(FIFO2);

}

/\*Client\*/

#include<stdio.h>

#include<unistd.h>

#include<sys/stat.h>

#include<fcntl.h>

#define FIFO1 "fifo1"

#define FIFO2 "fifo2"

#define PERMS 0666

char fname[256];

int main()

{

ssize\_t n;

char buff[512];

int readfd,writefd;

printf("Trying to Connect to Server..\n");

writefd = open(FIFO1, O\_WRONLY, 0);

readfd = open(FIFO2, O\_RDONLY, 0);

printf("Connected..\n");

printf("Enter the filename to request from server: ");

scanf("%s",fname);

write(writefd, fname, strlen(fname));

printf("Waiting for Server to reply..\n");

while((n=read(readfd,buff,512))>0)

write(1,buff,n);

close(readfd);

close(writefd);

return 0;

}

Output (Server)

[root@localhost CN Lab] ./s.o

Waiting for connection Request..

Connection Established..

Client has requested file alpha

[root@localhost CN Lab]Computer Networks Lab Manual

Output (Client)

[root@localhost CN Lab] ./c.o

Trying to Connect to Server..

Connected..

Enter the filename to request from server: alpha

Waiting for Server to reply..

This a demo of client server using Sockets

Just for trial.

Now End of file

**Viva Questions**

1.What are the different framing techniques?

Ans. Four framing methods

* + 1. character count
    2. flag bytes with byte stuffing
    3. starting and ending flags with bit stuffing

2.What is the drawback with character stuffing?

Ans. Incase of byte stuffing an Escape bytes(flag byte) is stuffed into the outgoing character stream before a flag byte in the data.

3.Express the following bit groups in polynomial form: 1101,

Ans.1001, 1111, 0101.

x3+x2+1 , x3+1 , x3+x2+x+1 , x2+1.

4. What types of errors can be detected using CRC method?

Ans. Single bit errors , Double errors and Burst errors.

5. What is the problem with distance vector routing?

Ans: Problem with the distance vector routing algorithm is called as count to infinity. It reacts rapidly to good news, but leisurely to bad news.

6. What is broadcasting?

Ans. Sending a packet to all destinations simultaneously is called as broadcasting.

7. Mention the formula for developing plain text from cipher text using RSA algorithm?

Ans. M= Cd mod n.

8.Can Wireshark Be Setup On A Cisco Router?

Ans : Wireshark is an executable. It can be setup on operating systems like windows and linux. It cannot be setup on a Cisco router , as it runs a proprietary operating system on which additional tools or software cannot be installed.

9.What is the use of NMAP tool?

Ans.Nmap, short for Network Mapper, is a free, open-source tool for vulnerability scanning and network discovery. Network administrators use Nmap to identify what devices are running on their systems,

discovering hosts that are available and the services they offer, finding open ports and detecting security risks

10. What are the four files on the NS2 simulator?

Ans.The four files include the .tcl file, the .awk file, the .tr file and the nam file.

11. What is the use of a tr file?

Ans.The tr file is mainly used to analyze our data. It is given as input to nam file.

12. What does $1,$2,..... indicate in the awk file?

Ans.The $1, $2,.... indicate the columns that represent different events in the ex3.tr file.

13.Which Wireshark filter Ccn be used to check all incoming requests to a Http Web Server?

Ans. HTTP web servers use TCP port 80. Incoming requests to the web server would have the destination port number as 80. So the filter tcp.dstport==80.

14. Which Wireshark filter can be used to monitor outgoing packets from a specific system on the network?

Ans. Outgoing packets would contain the IP address of the system as it’s source address.

So assuming that the IP address of the system is 192.168.1.2, the filter would be ip.src==192.168.1.2